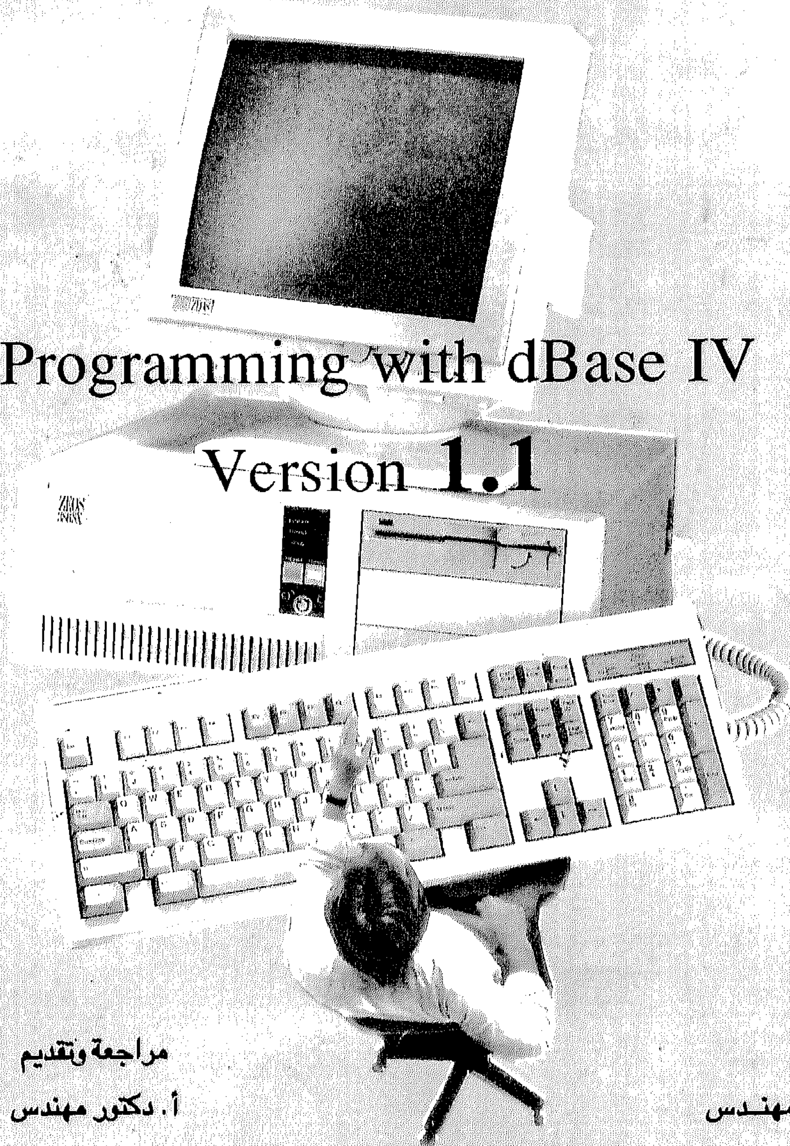
البرمجة بإستخدام قاعدة البيانات ٤

الاصدار المعدل (1.1)

الجزء الثالث



Programming with dBase IV

Version 1.1

مراجعة وتقديم

أ. دكتور مهندس

محمد صلاح الدين عباس

مهندس

على محمود بهلول

دار الكتب العلمية

للنشر والتوزيع

البرمجة باستخدام قاعدة البيانات ٤

الاصدار المعدل (1.1)

Programming with dBase IV
Version 1.1

الجزء الثالث

مهندس
علي محمود بهلول

مراجعة وتقديم
أ . دكتور مهندس
محمد صلاح الدين عباس

دار الكتب العلمية
للنشر والتوزيع

١٨ شارع السبع - ترعة السواحل إمبابه

بسم الله الرحمن الرحيم

﴿ وقل ربِ زدنى علما ﴾

صدق الله العظيم

الفصل الأول

مقدمة فى البرمجة

* مقدمة فى البرمجة باستخدام dBASE IV.
* طرق إنشاء البرنامج .
* التعامل مع نقطة الأوامر.
* إنشاء ملف البرنامج .
* بناء البرنامج .
* مكونات نموذج البرنامج الرئيسى.
* حفظ البرنامج .
* تعديل البرنامج .
* تنفيذ البرنامج .
* أنواع البيانات .
* العلامات والرموز المستخدمة .
* الشكل العام لاستخدام أوامر dBASE IV.

الفصل الأول

مقدمة فى البرمجة باستخدام dBASE IV.

أولاً : البرمجة باستخدام dBASE IV.

البرنامج هو مجموعة من التعليمات او الخطوات اللازمة لإنجاز مهام معينة ويتيح انجاز عدة مهام مختلفة مرة واحدة باستخدام أمر التشغيل (DO) وهذا بالطبع يختصر الوقت والجهد اللازمين لإنجاز نفس المهام بإصدار الأوامر متتابعة من خلال نقطة الأوامر (Dot Prompt) .

ثانيا : طرق إنشاء البرنامج فى dBase IV

يمكن إنشاء البرنامج بإحدى طريقتين ، هما :

١- باستخدام منتج التطبيقات (Application Generator) ، وذلك أسلوب سهل ولا يحتاج لخبرة سابقة فى البرمجة ليمكنك من كتابة برامج ، ولكنه يفتقر الى المرونة فى كتابة الأوامر ، وكذلك الى المرونة فى التحكم فى البرنامج ، وسيأتى شرح ذلك الأسلوب بالتفصيل فى الفصل المخصص لذلك .

٢- عن طريق كتابة الأوامر مباشرة من خلال شاشة معالج الكلمات (Word processor) الخاص بـ dBase IV أو من أى معالج كلمات آخر مثل (Word Star) .

ثالثاً : التعامل مع نقطة الأوامر

الوصول الى نقطة الأوامر (Dot prompt) ..

١- إجعل الحاسب فى وضع التشغيل (Power ON) ، وعندما تظهر الإشارة < C ؛ فإن لم تكن موجودة فى الدليل (Directory) الموجود به dBase IV ، عليك أن تصل اليه عن طريق كتابة أمر تغيير الدليل كما يلى :

. C > CD/dBase

٢- إكتب dBase واضغط مفتاح <ENTER> تظهر لك لوحات مركز التحكم (Control center) .

٣- للوصول الى نقطة الأوامر اضغط F10 ثم إختر (Exit) لتظهر لك قائمة فرعية ، إختر منها EXIT TO DOT PROMPT .

ستظهر لك نقطة يليها المؤشر فى أسفل يسار الشاشة فوق سطر الحالة (Status bar) الذى يعطى معلومات عن الملف المفتوح حاليا إن وجد .

ملحوظة : من الممكن الوصول مباشرة من نظام التشغيل (DOS) الى نقطة الأوامر وذلك بتعديل ملف Config. db ، وذلك بحذف السطر التالى :

COMMAND = ASSIST

* إدخال الأوامر .

تتم بكتابة الأوامر مباشرة والضغط على مفتاح <ENTER> ، ومن الممكن تعديل الأوامر قبل الضغط على مفتاح <ENTER> ، وذلك باستخدام مفتاح (Backspace) او باستخدام مفتاحى السهم لليمين (<—) ولليسار (—>) .

* الحصول على أوامر سبق إدخالها من خلال نقطة الأوامر .

يوجد فى dBase IV وسيط خازن (Buffer) يسمى (History Buffer) يخزن حتى ٢٠ أمرا ، ولإعادة الحصول على أمر سبق إدخاله ، إضغط (↑) ، لتحصل على الأمر السابق مباشرة ، وعند ضغط (↑) مرة أخرى تحصل على الأمر السابق لهذا الأمر ، وهكذا ..

ولإعادة تنفيذ أمر سبق تنفيذه ، إضغط (↑) عدد المرات اللازمة للوصول إليه واضغط <ENTER> .

* الحصول على بيانات توضيحية مساعدة من خلال نقطة الأوامر.

إكتب Help ، أو اضغط F1

وللحصول على بيانات مساعدة عن نقطة محددة إكتب (Help) اولاً ، متبوعة بالأمر الذى تريد الحصول على معلومات مساعدة عنه .

* الخروج من نقطة الأوامر

١- إضغط F2 للإنتقال الى مركز التحكم (Control center) او اكتب Assist واضغط <ENTER> للوصول الى مركز التحكم .

٢- إكتب Quit للخروج من dBase IV والإنتقال الى نظام التشغيل (DOS).

ملخّص مفاتيح الوظائف

المفــتاح	البـــــيان
F1	Help
F2	Assist
F3	List
F4	Dir.
F5	Display Structure
F6	Display Status
F7	Display Memory
F8	Display
F9	Append
F10	EDIT

رابعاً : إنشاء ملف البرنامج (Program file)

من نقطة الأوامر (Dot prompt) إكتب :

Modify Command <filename>

ومالم تكتب إسم الملف ، فسوف تظهر لك رسالة تطلب كتابة إسم الملف وتظهر لك شاشة معالج الكلمات الخاص بـ dBase IV . وبإمكانك استخدام أى معالج كلمات (Word processor) آخر ، بشرط أن تعدل فى ملف Config. db ، فإذا أردت مثلا أن تستخدم برنامج Word Star ، أضف هذا الأمر للملف : TEDIT = WS.

معالج الكلمات الخاص بـ dBase IV المخصص لكتابة ملفات البرامج من نوع (Word-wrap mode) .

Layout   Words   Go To   Print   Exit   2:11:40 pm
[......1......2......3......4......5......6......7......]

Program   C:/dbase/<NEW>   Line: 1   Col: 1   Num   Ins

شاشة معالج الكلمات الخاص بـ dBase IV

تذكـــــر :

١- يمكنك كتابة برنامج يحتوى حتى ٣٢,٠٠٠ سطرا كحد أقصى .

٢- يمكنك كتابة حتى ١٠٢٤ حرفا فى كل سطر أمر ، ويمكنك كتابة سطر الأمر على أكثر من سطر بإضافة العلامة (;) بعد آخر حرف فى السطر الأول .

٣- يمكنك كتابة ملف البرنامج من خلال نافذة منفصلة (Window) كما يلى: من نقطة الأوامر إكتب :

Modify command <PRG.file> Window <windowname>

٤- يمكنك نسخ (Copy) أو تحريك (Move) أجزاء من البرنامج كالتالى :

أ - إختر الجزء المراد نسخه أو تحريكه ، وذلك بوضع المؤشر Cursor فى بداية الجزء المراد نسخه أو تحريكه واضغط F6 .

ب - حرّك المؤشر لنهاية الجزء المراد نسخه أو تحريكه واضغط <ENTER> .

جـ - ضع المؤشر فى المكان الذى تريد النسخ او التحريك اليه ، ثم اضغط مفتاح F7 للتحريك ، أو مفتاح F8 للنسخ .

٥- يمكنك كتابة بعض التعليقات الخاصة بغرض التوضيح بوضع العلامة (*) فى بداية السطر إذا أردت سطرا كاملا أو وضع العلامة (&&) بعد أى سطر أمر ، ثم كتابة التعليق بعد هذه العلامة ، مع العلم بأن الحاسب لا يتعامل مع هذه التعليقات على أنها جزء من البرنامج .

٦- يمكنك زحزحة بعض الأسطر للداخل [الأسطر التى تخضع لشرط ما] ، مثل أسطر الأوامر الواقعة بين :

Dowhile .. ENDDO
IF .. ENDIF
SCAN .. ENDSCAN
DOcase .. ENDCASE

وذلك بضغط مفتاح (TAB) فى بداية أول سطر ، فتتم زحزحة بقية الأسطر تلقائيا بنفس المسافة .

يمكنك زحزحة سطر معين للخارج ،ذلك بضغط Shift-Tab .

خامساً : بناء البرنامج :

أ - البرمجة التركيبية (Moduler Programming) : هى مجموعة من الجزئيات المبرمجة (Modules) ، كل جزئ مبرمج منها يقوم بأداء مهمة محددة ، ويجمع كل هذه الجزيئات المبرمجة برنامجا واحدا يسمى البرنامج الرئيسى (Main program) او البرنامج المستدعى (Calling program) ، وهو يتحكم فى كل الجزيئات المبرمجة .

ويطلق على هذا النوع من البرمجة "البرمجة من القمة الى القاع" (Top-bottom programming) .

يسهل هذا النوع من البرمجة إنشاء البرامج لأنه يمكّن المبرمج من اختبار كل برنامج (Module) على حدة ، وكذلك يمكن المبرمج من إعادة استخدام أى جزئ مبرمج فى أجزاء متفرقة من البرنامج الكلى ، كما يمكن استخدامه فى برامج أخرى .. وهذا يقلل من الحيّز المشغول للبرنامج على القرص الممغنط ، ويزيد بالتالى من سرعة إنجازه .

ب - البرنامج الرئيسى (Main Program) : يتم فيه الآتى :

١- تجهيز الوسط المناسب لتقابل متطلبات البرنامج باستخدام(Set option).

٢- إنشاء حقول الذاكرة المتغيرة (Memory variables) .

٣- إنشاء القوائم (Menus) التى ترشد المستخدم عما يستطيع البرنامج أن ينجزه ويقابل متطلباته .

٤- يتحكم فى كل الجزيئات المبرمجة وهى البرامج الفرعية (Subprograms) او الإجراءات (Procedures) عن طريق استخدام الحلقة الدوارة الرئيسية (Dowhile ..ENDDO) .

٥- إنهاء البرنامج وغلق جميع الملفات وإعادة تهيئة الوسط المناسب

للوضع المسبق له.

سادساً: مكونات نموذج البرنامج الرئيسى

١- إمسح الشاشة عن طريق الأمر Clear .

٢- إغلق جميع الملفات المفتوحة باستخدام الأمر Close database .

٣- إمسح الحقول المتغيرة Memory vriables التى قد تكون فى ذاكرة الحاسب ، بالأمر Clear all .

٤- جهّز الوسط المناسب (Environment) للبرنامج باستخدام (Set options)

Set talk off
Set status off
Set escape off

٥- إفتح ملفات قاعدة البيانات والملفات المرتبطة بها ..

Use <filename>

Set view to <filename>

٦- إنشئ الحلقة الدوّارة الرئيسية التى تتحكم فى كل الجزيئات البرمجية ..

Dowhile .T.
ENDDO

٧- إنشئ القوائم (Menus) التى تقابل متطلبات البرنامج واجعلها فى وضع التنفيذ ..

Activate Popup <popup name>

٨- إنتظر اختيار المستخدم لأحد اختيارات القائمة .

٩- عند اختيار المستخدم لخيار ما يقوم البرنامج الرئيسى باستدعاء الجزئية البرمجية (إما برنامج فرعى ، أو إجراء معيّن) ، وينفذه .

١٠- عند إتمام تنفيذ الإختيار ، تعود شاشة القائمة الرئيسية مرة أخرى حتى يتمكن المستخدم من اختيار أى من الإختيارات الموجودة بالقائمة .

١١- إذا أراد المستخدم إنهاء العمل مع البرنامج والخروج ، فإن أحد إختيارات القائمة يجب أن تكون مخصصة لذلك والتى تنفذ الأمر (Exit) الذى يمكّنه من الخروج من الحلقة الدوارة الرئيسية .(Dowhile .T. .. ENDDO)

سابعاً : حفظ البرنامج (Program File Saving)

بعد كتابة آخر أمر فى البرنامج تستطيع حفظ ملف البرنامج ، بضغط Ctrl-End.

ثامنا: تعديل البرنامج :

من نقطة الأوامر Dot Prompt . إكتب :

. Modify command <filename>

وتستطيع الإنتقال بين أجزاء البرنامج كما يأتى :

١- باستخدام مفتاحى السهم لأسفل (↓) والسهم لأعلى (↑) ، ويتم التحرك بمقدار سطر واحد ، كما يمكن التحرك خلال السطر باستخدام مفتاحى السهمين لليمين ولليسار ، ويمكن التعديل باستخدام مسطرة المسافات (Space bar) لحذف حرف فى الإتجاه الأمامى ، ومفتاح (Back space) لحذف حرف فى الإتجاه العكسى ، ويمكن حشر (Insert) حرف أو كلمة فأكثر فى حالة أن يكون المفتاح (Insert) فى الوضع (ON) او الكتابة فوق الكتابة الموجودة فى حالة المفتاح (Insert) فى الوضع (OFF) .

٢- يمكن الحركة بمقدار شاشة كاملة باستخدام مفتاح (PgDn) للتحرك بمقدار صفحة للأمام ، ومفتاح (PgUp) للتحرك بمقدار صفحة للخلف.

ولحفظ التعديل إضغط Ctrl-End للحفظ والخروج الى نقطة الأوامر (Dot prompt) .

ولإلغاء التعديل والخروج ، إضغط مفتاح (ESC)

تاسعا : تنفيذ البرنامج :

يتم تنفيذ البرنامج باستخدام الأمر Do : الصورة العامة هى

. Do <filename>

مثال : Do inventory

ينفذ البرنامج Inventory .

ويمكن تنفيذ البرنامج مباشرة بمجرد تحميل dBase IV وذلك بإضافة السطر التالى الى ملف Config.db :

COMMAND=DO inventory

ويمكن كذلك تشغيل البرنامج مباشرة من نظام التشغيل (DOS) كما يلى:

C > dBase inventory

فيتم تشغيل البرنامج بعد عدة ثوان أو مباشرة بضغط مفتاح <ENTER> .

عاشراً : أنواع البيانات المستخدمة فى dBase IV :

١- البيانات الحرفية ، ويرمز لها بالرمز <Exp C> .
٢- البيانات العددية ، ويرمز لها بالرمز <Exp N>.
٣- البيانات العددية ذات الطابع العلمى (Floating) ، ويرمز لها بالرمز <Exp F> .
٤- البيانات التاريخية ، ويرمز لها بالرمز <Exp D> .
٥- البيانات المنطقية ، ويرمز لها بالرمز <Condition>.

حادى عشر : العلامات والرموز المستخدمة فى dBase IV

١- العلامات الحسابية (Math. operators) ..

+ الجمع
- الطرح
* الضرب
/ القسمة
** او ^ رفع الأس
() للتجميع

٢- علامات العلاقات (Relational operators)

< أصغر من
> أكبر من
= يساوى
≠ أو <> لا يساوى
<= أصغر من أو يساوى
>= أكبر من أو يساوى
$ لمقارنة تعبير بتعبير آخر أو بجزء منه.

٣- العلامات المنطقية (Logical opertors)

.AND. تستلزم تحقق الشرطين
.OR. تستلزم تحقق أحد الشرطين
.NOT. تستلزم عدم تحقق الشرط
() للتجميع

ثانى عشر: الشكل العام لاستخدام أوامر dBase IV ..

[<command verb>], [<expression list >], [<scope>],
[FOR <condition>], [WHILE <condition>],
[To file <filename> / TO PRINTER / TO ARRAY <array list> / To <memvar>].

١- <command verb> يقصد بها أمر dBase IV .

٢- [ ] ما يكتب بين الأقواس المربعة يعنى أنه اختيارى .

٣- < > ما يكتب بين الأقواس الحادة يعنى أنه إجبارى ، ويجب أن يعطى قيمة .

٤- / ما يجيئ بعد الخط المائل (Slash) يعنى أنه اختيار مرادف لما قبل الخط المائل .

٥- <list > مجموعة من العناصر ، ويتم الفصل بين كل عنصر وآخر باستخدام الفاصلة (Comma) .

٦- <expression list > تعبير أو أكثر يفصل بين كل تعبير وآخر بواسطة فاصلة .

٧- <Scope > لتحديد عدد السجلات التى يشملها تنفيذ أمر dBase ويكون أحد أربعة إختيارات ..

أ - Record <n> لتحديد سجل محدد بواسطة رقمه يطبق عليه أمر dBase .

ب - next <n> لتحديد عدد (N) سجلا يطبق عليهم الأمر وتبدأ من السجل الحالى .

جـ - All لجعل أمر dBase يطبق على جميع سجلات ملف قاعدة البيانات .

د - Rest لجعل الأمر يطبق على جميع سجلات الملف التى تبدأ من السجل الحالى وحتى آخر الملف .

٨- FOR <condition> لتحديد شرط ما ليطبق أمر dBase على السجلات التى تحقق هذا الشرط فقط .

٩- While <condition > تستخدم لتنفيذ الأمر على السجلات طالما تحقق الشرط ، ويتم إيقاف تنفيذ الأمر عند أول سجل لا يحقق هذا الشرط .

١٠- [To ..] للتحكم فى الجهة التى سيرسل اليها نتيجة تنفيذ أمر dBase ، وتكون هذه الجهة أحد أربعة اختيارات ..

أ - حقول ذاكرة متغيرة (Memvars) .
ب - الطابعة Printer .
جـ - ملف File .
د - مصفوفة ذاكرة Array .

الفصل الثانى

تجهيز وتهيئة الوسط

* كيفية تغيير الوضع المسبق للوظائف ...

Set Century - Set Carray - Set Catalog - Set Bell - Set Echo - Set Date - Set Color - Set Clock - Set Manage - Set Escope - Set Talk - Set Status - Set Safety

الفصل الثانى

تجهيز وتهيئة الوسط SETUP ENVIRONMENT
(Set Options)

توجد بعض الوظائف المسبق ضبطها فى dBase IV والتى قد يلزم فى بعض الأحيان تغييرها لتناسب متطلبات البرنامج الذى تقوم بإنشائه . ويمكنك تغيير هذا الضبط المسبق (Default) بإحدى طريقتين :

١- عن طريق التعديل فى ملف Config.db.

مثال :

الوضع المسبق لسطر (Status bar) هو الظهور أسفل الشاشة ، ولكنك قد ترغب فى أن لا يظهر أثناء تنفيذ البرنامج ، فيمكنك تعديل السطر Status = ON الى Status = OFF

٢- عن طريق الأمر SET

وهذا يجعلك تغير الوضع المسبق مؤقتا ، أى أنه يُلغى عند الخروج من dBase IV ، أو عن طريق تغيير أمر SET مرة أخرى .

وسنستعرض بعض هذه الوظائف المستخدمة فى تهيئة الوسط فى البرنامج .

١ Set Bell

يتيح تغيير التنغيم الصوتى للجرس Set bell ON/off

الصورة العامة : Set bell To [<freq>, <duration>] .

يمكن تغيير التردد (Frequency) ، والفترة (Duration) ؛

الوضع المسبق هو ON .

Frequency = 512 Hz التردد .

Duration = 2 tiks الفترة .

مثال :

لتغيير التنغيم الصوتى ؛ Set Bell To 400,2 .

٢ Set Catalog

تستخدم لإنشاء أو فتح ملف كتالوج ؛ عند فتح الكتالوج تضاف اليه الملفات التى ستفتحها او التى ستنشئها للملف الكتالوج .

<u>الصورة العامة</u> : Set Catalog To <filename>

Set Catalog on/OFF

الوضع المسبق (Default) هو OFF .

٣ Set Carry

تنسخ بيانات السجل الأخير الى السجل الجديد عند استخدام الأمر Append أو الأمر INSERT فى حالة أن يكون الوضع (ON) .

<u>الشكل العام</u> : Set Carry on/FF

الوضع المسبق (OFF) .

٤ Set Century

تتيح إدخال القرن ضمن التاريخ ، وبذلك تكون الأرقام المخصصة للسنة أربعة أرقام وليست رقمان .

الشكل العام : Set Century on/OFF

الوضع المسبق (OFF) .

مثال :

```
? DATE ( )
12/10/89

Set Century ON
? DATE ( )
12/10/1989
```

٥ Set Clock

للتحكم فى ظهور الساعة أو عدم ظهورها ، وكذلك فى موقع ظهورها على الشاشة.

١ - Set clock to [< row > , < clumn

لتحديد موقع ظهور الساعة على الشاشة .

٢ - Set clock on/OFF

لإظهار الساعة على الشاشة أو عدم إظهارها .

مثال :

```
Set clock ON
Set clock to 1,68
```

لإظهار الساعة فى الصف (١) والعمود (٦٨) على الشاشة .

٦ Set Color

تتيح لك اختيار الألوان التى تناسب رغباتك ، سواء بالنسبة للشاشات الملونة او الأحادية الألوان .

الشكل العام : Set Color ON/OFF

Set color To [[<Standard>][,[<enhanced>][,[<parameter>]
[,[<Background>]]]]

Set color OFF تحول الشاشة الملونة الى شاشة غير ملونة (monochrome) .

STANDARD تعنى ألوان الشاشة الأمامية والخلفية .

Enhanced تعنى ألوان الشاشة العكسية : الأمامية والخلفية .

مثال -١ :

Set color to W/N, N/W, N

للكتابة بلون أبيض على أسود ، والشاشة العكسية باللون الأسود على الأبيض وخلفية الشاشة سوداء .

مثال -٢ :

Set color to GR+/B, G/W, N

للكتابة بلون أصفر على خلفية زرقاء ، وألوان الشاشة العكسية الكتابة باللون الأخضر على اللون الأبيض ، والخلفية باللون الأسود .

جدول الألوان

الرمز	اللون
N	أسود
B	أزرق
R	أحمر
W	أبيض
G	أخضر
BG	أخضر فاتح
N+	رمادى
RB	أحمر فاتح
GR	بنى
GR+	أصفر
X	تخفى الكتابة

Set Date ٧

تحدد شكل ظهور التاريخ :

الشكل العام : Set date [To] AMERICAN/ansi / British / French / German / Italian / Japanese / mdy / dmy / ymd

الوضع المسبق (Default) له AMERICAN .

جدول أنماط كتابة التاريخ

الرمــز	شكل التاريخ
AMERICAN	ش ش / ى ى / س س [MM / DD / YY]
ANSI	س س . ش ش . ى ى [YY . MM . DD]
BRITISH/FRENCH	ى ى / ش ش / س س [DD / MM / YY]
GERMAN	ى ى . ش ش . س س [DD . MM . YY]
ITALIAN	ى ى – ش ش – س س [DD - MM - YY]
JAPANESE	س س / ش ش / ى ى [YY / MM / DD]
USA	ش ش – ى ى – س س [MM - DD - YY]
MDY	ش ش / ى ى / س س [MM / DD / YY]
DMY	ى ى / ش ش / س س [DD / MM / YY]
YMD	س س / ش ش / ى ى [YY / MM / DD]

ملحوظة : ى ى [DD] تعنى رقم اليوم
ش ش [MM] تعنى رقم الشهر
س س [YY] تعنى رقم السنة

٨ Set Default

يتيح التحوّل الى مشغل وحدة (Drive) آخر غير المشغل الحالى .

الشكل العام : . Set default To <drive> [:]

فمثلا إذا أردت أن تعدل فى ملف موجود على قرص ممغنط فى مشغل الوحدة A ، إكتب الآتى :

.Set default to A:
.Use <file>
.Edit

٩ Set Echo

تعرض سطر الأمر (Command line) أثناء تنفيذه أو تطبعه على الطابعة .

الشكل العام : Set Echo on/OFF

الوضع المسبق هو (OFF).

١٠ Set Escape

الشكل العام : Set ESCAPE ON/off

الوضع المسبق هو ON

عندما يكون الوضع على ON ويُضغط مفتاح (ESC) أثناء تنفيذ البرنامج ، فإن ذلك يؤدى للخروج من البرنامج وتعرض الرسالة التالية :

*** INTERRUPTED ***

Cancel, Ignor, Suspend ? (C, I, or S)

لذلك يُجعل الضبط على الوضع (OFF) لضمان عدم توقف البرنامج إذا قام المستخدم بالضغط خطأ على مفتاح (ESC) .

ويمكن استخدامها فى الوضع (ON) على أن يستخدم الأمر ON ESCAPE فى البرنامج.

١١ Set Exact

الشكل العام : Set exact ON/OFF

الوضع المسبق هو OFF

عندما تكون فى الوضع OFF فإنه لا يشترط التطابق التام فى المقارنة بين تعبيرين ، بل يكفى أن يشتركا فى بعض الحروف .

مثال :

```
.Set exact off
.Use student
.Display all For name = 'AHMED'
```

يؤدى الى إظهار جميع الأسماء فى ملف قاعدة البيانات (Student) التى يكون الأسم الأول منها = AHMED بغض النظر عن الإسم الثانى أو الثالث .
وعندما تكون فى الوضع (ON) فإنه يشترط التطابق التام بين التعبيرين (أى أن يكون لهما نفس الطول) ، وفى هذه الحالة فإنه يستلزم كتابة الإسم كاملا أى مشتملا على الإسم الثانى والثالث ، وإلاّ فلن يتم ظهور أى سجل .

١٢ Set Function

مفاتيح الوظائف (Function keys) المتاحة فى dBase IV عشرة ؛ من F1 الى F10 ، والوضع المسبق للوظائف كما يلى :

F1 Help;	F6 Display Status;
F2 Assist;	F7 Display Memory;
F3 List;	F8 Display;
F4 Dir;	F9 APPEND;
F5 Display Structure;	F10 EDIT;

ومن الممكن تغيير الوضع المسبق لجميع الوظائف فيما عدا (F1 Help) ، وتستطيع تحديد حتى ٢٣٨ حرفا لكل وظيفة .

مثال :

تستطيع تغيير الوضع المسبق لمفتاح الوظيفة (F3) من (;List) الى (Modify command;) ؛ كما يلى :

```
Set function F3 To "Modify command"
```

١٣ Set Message

تتيح عرض رسالة أسفل الشاشة (فى السطر ٢٣) فى حالة أن يكون (Status) فى الوضع (ON) .

الشكل العام : Set Message To [<expc>] .

١٤ Set Safety

تمنع المستخدم من الكتابة فوق ملف موجود وذلك بعرض رسالة تطلب من المستخدم التأكد من أنه يريد الكتابة فوق ملف موجود .

الشكل العام : Set Safety ON/OFF

الوضع المسبق هو (ON)

فى الوضع (ON) ، إذا أنشئ ملف له نفس إسم ملف موجود فعلا ، فإن الرسالة التالية سوف تظهر :

File already exists
Overwrite Cancel

Overwrite : معناها أنك تريد الكتابة على الملف القديم ، وذلك يعنى فقد بيانات الملف القديم .

Cancel : تتيح لك باختيارها فرصة لإدخال إسم آخر يختلف عن الإسم القديم (المسجل سلفا) .

١٥ Set STATUS

تحدد إن كان سيتم إظهار سطر الحالة الراهنة (Status bar) أسفل الشاشة أم لا .

الشكل العام: Set Status ON/OFF

الوضع المسبق

وسطر الحالة الراهنة (Status bar) يعرض الأمر المستخدم حالياً وإسم الملف المفتوح ورقم السجل الحالى بالنسبة لمجموع سجلات الملف .

Set Talk [١٦]

تحدد ما إذا كان سيتم إظهار معلومات أثناء تنفيذ بعض الأوامر أم لا (سواء بعرضها على الشاشة أو بطبعها على الطابعة) .

الشكل العام : Set talk ON/OFF

الوضع المسبق هو (ON)

تستخدم فى البرنامج فى الوضع (OFF) وذلك لعدم إظهار معلومات تنفيذ بعض الأوامر ؛ مثل : pack, append from, Store, Copy.

مثال :

عند كتابة الأمر ...

STORE "AHMED" TO MNAME

سواء من نقطة الأوامر (Dot prompt) او من خلال برنامج ، فإن السطر

.AHMED.

سيظهر ، ويعنى أنه قد تم تنفيذ الأمر ، وهو إنشاء حقل ذاكرة متغير إسمه MNAME وقد وضع فيه "AHMED".

وللرغبة فى عدم ظهور هذا السطر يتم وضع الأمر (Set talk) فى الوضع (OFF) كما يلى :

.Set talk OFF
.Store "AHMED" To MNAME

لاحظ عدم ظهور السطر الذى يفيد بأن الأمر Store قد نُفذ .

الفصل الثالث

إستخدمام حقول الذاكرة المتغيرة

* تعريف حقول الذاكرة المتغيرة.
* طرق إنشاء حقول الذاكرة المتغيرة.
* مصفوفة الذاكرة - ملء عناصرها - نسخ البيانات من مصفوفة الذاكرة الى ملف قاعدة البيانات - إستخدام المصفوفة فى البرنامج.
* الأمر PRIVATE.
* الأمر PUBLIC.
* حفظ حقول الذاكرة المتغيرة على القرص الممغنط.
* حذف محتويات الذاكرة.

الفصل الثالث

إستخدام حقول الذاكرة المتغيرة
MEMORY VARIABLE

تعريف حقول الذاكرة المتغيرة Memvars

هى مواقع فى الذاكرة مخصصة للاحتفاظ ببعض البيانات مؤقتا لاستخدامها فى التحكم فى تتابع إجراء العمليات داخل البرنامج ، وهى تختلف عن حقول الذاكرة (Memo fields) داخل ملف قاعدة البيانات فى أنها تحتفظ بالبيانات بشكل مؤقت ، فى حين أن حقول الذاكرة بالملف تحتفظ بها بشكل دائم .

ملحوظة: حقول الذاكرة المتغيرة (Memvars) تكون مخزنة داخل الذاكرة الرئيسية للحاسب (RAM) وتُفقد بمجرد الخروج من dBase IV أو بإطفاء الحاسب (Power OFF) وذلك ما لم تحفظ داخل ملفات على القرص الممغنط (Disk) .

من الممكن استخدام حتى ١٥٠٠٠ حقلا متغيرا كحد أقصى ، وذلك بتغيير الوضع المسبق (Default) فى ملف Config.db .

طرق إنشاء حقول الذاكرة المتغيرة

باستخدام أى من الأوامر الآتية :

١- Store.
٢- Accept .
٣- Input .
٤- Wait .
٥- Declare .

١- الأمر Store

ينشئ حقل ذاكرة متغير واحد أو أكثر أو يملأ عناصر مصفوفة الذاكرة (Memory Array) التى سبق إنشاؤها بواسطة الأمر Declare .

الإستخدام العام :

STORE <exp> TO <memvar list >/<array element list>

نوع البيانات يتحدد تلقائيا من البيانات المراد تخزينها داخل حقل الذاكرة المتغيرة .

أ - .STORE 18.50 TO NUM

يحتوى حقل الذاكرة NUM الآن على الرقم ١٨,٥٠

ب - يمكنك أن تنشئ أكثر من حقل ذاكرة متغير واحد من نفس النوع على أن يفصل كل منها عن الآخر بفاصلة (Comma) ، هكذا :

```
.Store { /  / } to mdate1, mdate2, mdate3
  / /
```

جـ - يمكنك أن تخزن رسالة تذكير أو جملة داخل حقل ذاكرة ؛ هكذا :

```
.Store "IS THIS CORRECT?"  To Mess
IS THIS CORRECT?
.? MESS
IS THIS CORRECT?
```

د - يمكنك أن تخزن العلامات المنطقية (.Y.), (.N.), (.T.), (.F.) هكذا :

```
.Store  .T.  to result
.T.
.? RESULT
.T.
```

طريقة أخرى لإنشاء حقول الذاكرة المتغيرة ..

وذلك بكتابة اسم الحقل المتغير يسار علامة (=) ، ثم البيانات التى تخزنها داخل الحقل يمين علامة (=) ؛ هكذا :

أ - .NUM = 18.50

18.50

ب - .Result = .T.

.T.

. ? Result

جـ - .MDate1 = { / / }

/ /

يستخدم الأمر Store كذلك لملء عناصر مصفوفة الذاكرة ؛

```
.Declare M[3,3]
.Store 2  To M[1,1]
      2
. ? M[1,1]
      2
```

يخزن الرقم (٢) فى العنصر الذى موقعه الصف الأول والعمود الأول للمصفوفة (M) .

مثال ٣-١:

فى برنامج مراقبة المخزون ، نفترض أن البرنامج يعطى إمكانية للمستخدم أن يستبدل بيانات أى صنف موجود داخل ملف قاعدة البيانات ، ببيانات أخرى يُدخلها المستخدم ، فإنه يستفسر من المستخدم أولاً عن رقم الصنف (ويجب أن يكون الملف مفهرساً على رقم الصنف) ، ثم يقوم بتخزين ما يدخله المستخدم داخل حقل ذاكرة متغير ، وبعد الوصول الى سجل الصنف يستحث المستخدم على أن يُدخل البيانات التى يريد أن يستبدل بها البيانات الموجودة بحقل ما فى السجل الحالى ، ثم يقوم بتخزينها داخل حقل ذاكرة متغيرة ، ويقوم باستبدال محتويات الحقل بمحتويات حقل الذاكرة المتغير.

```
1  CLEAR
2  SET TALK OFF
3  USE STORE ORDER NO
4  DO WHILE .T.
5  INPUT  "ENTER THE  ITEM NUMBER (0 TO EXIT):"   TO  MNO
6  IF MNO=0
7  EXIT
8  ENDIF
9  SEEK MNO
10  IF .NOT. FOUND ( )
11  CLEAR
12  @ 10,10 SAY "THIS ITEM IS NOT EXIST.. PLEASE RE-ENTER:"
13  LOOP
14  ENDIF
15  INPUT  "ENTER NEW RE-ORDER LEVEL:" TO MRLVL
16  REPLACE REORDER WITH MRLVL
17  ENDDO
18  RETURN
```

١- فى سطر (٥) يتم إنشاء حقل ذاكرة متغير (MNO) ليخزن به رقم الصنف .

٢- من سطر (١٠) الى سطر (١٤) حلقة شرطية تعنى أنه فى حالة عدم الوصول الى السجل الذى يحتوى على رقم الصنف الذى أدخله المستخدم فإنه يتم عرض رسالة تفيد بعدم وجود هذا الصنف ثم الإنتقال الى السطر (٥) .

٣- السطر (١٥) لتخزين البيانات التى يُدخلها المستخدم بداخل حقل ذاكرة متغير (MRLVL) .

٤- السطر (١٦) لاستبدال محتويات الحقل (Record) بمحتويات حقل الذاكرة المتغير (MRLVL) .

٢- الأمر Wait

ينشئ حقل ذاكرة متغير ، كما أنه يستخدم لإيقاف تنفيذ البرنامج بصفة مؤقتة حتى يضغط المستخدم أى مفتاح .

<u>الإستخدام العام</u> : Wait [<prompt >][To <memvar >]

مثال ٣-٣:

```
CLEAR
SET TALK OFF
DO WHILE  .T.
? "                    SAMPLE  MENU"
? "         A= EDIT"
? "         B= APPEND"
? ".        C= EXIT"
?
WAIT "ENTER YOUR CHOICE:" TO MCH
DO CASE
CASE UPPER (MCH)="A"
DO  PROG1
CASE UPPER (MCH)="B"
DO  PROG2
CASE UPPER (MCH)="C"
EXIT
ENDCASE
ENDDO
RETURN
```

فى هذا المثال يتم إنشاء حقل ذاكرة متغير (MCH) وذلك نتيجة لاستخدام الأمر (Wait) . ويقوم حقل الذاكرة المتغير هذا بتخزين ما يدخله المستخدم (من حرف واحد) ، وبناءً على هذه القيمة المخزنة داخل (MCH) يتم التفريع (Branching) باستخدام (Do case .. End case) .

وعند تنفيذ هذا البرنامج يتم عرض القائمة الموضحة بالشكل التالى :

```
                    SAMPLE MENU

         A = EDIT
         B = APPEND
         C = EXIT

ENTER  YOUR  CHOICE :
```

٣- الأمر Accept

يستحث المستخدم على أن يدخل بيانات معينة (من النوع الحرْفى) ويقوم بتخزينها داخل حقل ذاكرة متغير.

الإستخدام العام : ACCEPT [<prompt >] To <memvar >

البيانات التى يدخلها المستخدم تكون من النوع الحرفى حتى إذا أدخل المستخدم أرقاما فإنها تعامل على أنها حروف .

مثال :

نتيجة للأمر .. ACCEPT "ENTER THE ITEMNAME:" To MNAME.

تظهر على الشاشة الرسالة التالية : ENTER THE ITEM NAME:

وتبقى حتى يُدخل المستخدم إسم الصنف (Itemname) ويضغط مفتاح <ENTER> ، حينئذ سيقوم الحاسب بتخزين اسم الصنف الذى أدخله المستخدم داخل حقل ذاكرة متغير اسمه (MNAME) .

٤- الأمر Input

يقوم بحث المستخدم على إدخال بيانات ويقوم بتخزينها دا.خل حقل ذاكرة متغير.

الإستخدام العام : INPUT [<prompt >] To <memvar >

نوع البيانات التى يدخلها المستخدم تحدد نوع حقل الذاكرة المتغير إن رقمياً (Nmeric) أو حرفياً (Character) او تاريخاً (Date) او منطقياً (Logical) .

مثال :

نتيجة للأمر ..

INPUT "PLEASE ENTER THE ITEM NUMBER:" To MNO

تظهر على الشاشة الرسالة التالية :

PLEASE ENTER THE ITEM NUMBER

وتبقى حتى يُدخل المستخدم رقم الصنف (Item number) ليقوم بتخزينه داخل حقل ذاكرة رقمى اسمه (MNO) .

ملحوظة : إستخدام الأمر ACCEPT والأمر INPUT يعطيان مرونة كبيرة للبرنامج ، إذ أنهما يتيحان إجراء عمليات معينة على ما يدخله المستخدم من بيانات ، وليس على بيانات ثابتة موجودة داخل قواعد البيانات ، وهذا يعطى مدى واسع من المرونة بين البرنامج والمستخدم .

٥- الأمر Declare

لإنشاء المصفوفات مختلفة الأبعاد .

مصفوفة الذاكرة ARRAY

أ - تعريف المصفوفة : هى مجموعة من القيم المرتبة فى صفوف وأعمدة ؛ المصفوفة الأحادية البعد (One-dimensional-array) ذات صف واحد . المصفوفة الثنائية البعد (Two-dimensional-array) ذات أكثر من صف واحد ؛ كل صف من المصفوفة يمثل سجلاً (Record) فى ملف قاعدة البيانات ، بينما كل عمود فى الصف من المصفوفة يمثل حقلا (Field) فى ملف القاعدة ؛ لذا فإنه من الممكن أن تستخدم المصفوفة فى تخزين السجلات (Records) مؤقتاً قبل إضافتها للملف قاعدة البيانات.

	1	2	3	4
1	A	B	C	D
2	E	F	G	H
3	G	K	L	M

من الممكن تحديد عنصر معين من عناصر المصفوفة وذلك بكتابة إسم المصفوفة ثم رقم الصف ورقم العمود الذى يقع فيه العنصر ؛ فمثلا ، الحرف (H) من المصفوفة التى اسمها (M) يمكن تحديده كما يلى :
. M [2, 4]

ب - إنشاء مصفوفة الذاكرة (Array) :
تنشأ مصفوفة الذاكرة الأحادية البعد او الثنائية بالأمر Declare .
الإستخدام العام : Declare <definition list > .

مثال :

لإنشاء مصفوفة أحادية البعد : Declare M1 [10]

ولإنشاء مصفوفة ثنائية : Declare M2 [8,3]

ولإنشاء أكثر من مصفوفة : Declare L1 [8], L2 [6,2]

جـ – ملء المصفوفة Filling the array :

توجد ثلاث طرق لملء المصفوفة ..

١- باستخدام الأمر Store

يمكن ملء أى عنصر من عناصر المصفوفة بالبيانات عن طريق تحديد موقعه بتحديد رقم الصف ورقم العمود الذى يقع فيه العنصر .

مثال :

STORE "Ahmed" to mname [3,2]

يخزن "Ahmed" فى العنصر الذى موقعه الصف الثالث والعمود الثانى من المصفوفة (المسماة mname) .

٢- بأن تكتب إسم المصفوفة متبوعة بموقع العنصر يسار علامة يساوى (=) ، وتكتب البيانات التى تريد ملء عنصر المصفوفة به يمين علامة (=) .

مثال :

MNAME [4,2] = "KHALED"

يقوم بملء العنصر الذى موقعه الصف الثانى من العمود الرابع بالإسم (Khaled)

٣- باستخدام الأمر Copy To Array

يقوم بملء المصفوفة بمحتويات سجل (Record) او أكثر من ملف قاعدة بيانات .

الصورة العامة :

Copy to array <array name > [Fields <fields list >] [<scope >]
[FOR <condition >][while <condition >]

لكل سجل (Record) فى ملف قاعدة البيانات ، فإن أول حقل (Field) يخزن فى العمود الأول ، وثانى حقل يخزن فى العمود الثانى ، .. وهكذا .

وبذلك يصبح كل سجل فى الملف ممثلا لصف (فى المصفوفة) .

إذا كان عدد حقول (Fields) ملف قاعدة البيانات أكثر من عدد أعمدة المصفوفة ، فإن هذا العدد الزائد من الحقول لا يخزن داخل المصفوفة .

وإذا كان عدد أعمدة المصفوفة فى ملف قاعدة البيانات أكثر من عدد حقول الملف ، فإن عناصر هذه الأعمدة الزائدة لا تتغير .

ملحوظة : حقول الذاكرة (Memo fields) لا يمكن نقلها الى مصفوفة الذاكرة.

مثال :

```
Use master
Declare MARRAY [5,2]
Copy to array MARRAY for salary = 1000 while name = ;
"SAAD"
```

يقوم بإضافة سجلات من ملف قاعدة البيانات (Master) الى المصفوفة (MARRAY) التى تحقق شرط Salary = 1000 ، وطالما يكون الإسم Name = "Saad" .

د – نسخ البيانات من مصفوفة الذاكرة الى ملف قاعدة البيانات :

يضيف الأمر APPEND FROM ARRAY سجلات (Records) لملف قاعدة بيانات من بيانات مصفوفة ذاكرة .

الشكل العام :

```
APPEND FROM ARRAY <array name>[for <condition >]
```

تنسخ محتويات أول عمود لكل صف فى المصفوفة الى أول حقل (Field) فى الملف ، وثانى عمود يُنسخ فى ثانى حقل فى الملف ، وهكذا ..

فإذا وجد أن عدد أعمدة المصفوفة أكبر من عدد حقول البيانات فى ملف قاعدة البيانات ، فإن هذه الأعمدة الزائدة تهمل . وإذا وجد أن عدد حقول البيانات لملف قاعدة البيانات أكبر من عدد أعمدة المصفوفة ، فإن هذه الحقول تبقى بدون تغيير .

عند استخدام For <cond > فإنه يجب أن يتحقق هذا الشرط المحدد قبل إضافة أى صف من المصفوفة لملف قاعدة البيانات ، وإذا لم يتحقق الشرط فإن هذا الصف يهمل (Skipped) ، ويتم التعامل مع الصف الذى يليه.

ملحوظة : يجب أن يتم إعلان المصفوفة أولا باستخدام الأمر Declare ثم تخزن المعلومات فى عناصر المصفوفة قبل أن يمكنك استخدام الأمر APPEND .

مثال :

```
Declare MNAME [2,2]
MNAME[1,1]="AHMED"
MNAME[1,2]=100
MNAME[2,1]="KHALID"
MNAME[2,2]=1500
Use master
APPEND from ARRAY MNAME
```

هـ - إستخدام المصفوفة فى البرنامج :

تستخدم لاستقبال البيانات التى يدخلها المستخدم لسجلات الملف (Records) وعندما يتم التأكد من صحتها تنسخ لسجلات ملف قاعدة البيانات .

الأمر Private

يمكّنك من إنشاء حقول ذاكرة متغيرة فى برنامج فرعى لها نفس أسماء حقول متغيرة فى البرنامج الرئيسى أو التى سبق إعلانها عامة (Public) دون أن تتأثر محتوياتها بأى تغيير يطرأ على حقول الذاكرة التى لها نفس الإسم فى البرنامج الرئيسى .

الصورة العامة :

```
PRIVATE <memvar list >
PRIVATE ALL [like/except <skeleton >]
```

أمثلة :

١- PRIVATE MNAME, MSAL ، تعلن الحقلين Msal, Mname بالإعلان "خاص" Private.

٢- PRIVATE ALL like * MT ، تعلن الحقول التى أسمائها تبدأ بحرفى MT بالإعلان "خاص" (PRIVATE)

٣- PRIVATE AL EXCEPT MR?? ، تعلن الحقول التى أسماءها لا تبدأ بحرفى MR بالإعلان "خاص" (PRIVATE) .

٤- PRIVATE ALL ، تعنى أن كل الحقول المتغيرة (Memvar) المستخدمة داخل هذا البرنامج (Private) .

مثال :

إدخال بيانات الى ملف قاعدة البيانات (Test .dbf) - بنية الملف موضحة فى شكل (٣ - ١) - عن طريق استخدام مصفوفة ذاكرة (Array) كوسيط لتخزين البيانات التى يدخلها المستخدم ، وبعد التأكد من صحتها يتم استبدال محتويات حقول الملف بمحتويات عناصر المصفوفة .

Layout   Organize   Append   Go To   Exit   1:16:07 am

Bytes remaining: 3960

Num	Field Name	Field Type	Width	Dec	Index
1	NUM	Numeric	6	0	Y
2	NAME	Character	12		N
3	ADD	Character	15		N
4	TEL	Numeric	7	0	N

Database   C:/dbase/TEST   Field 1/4   Caps

Enter the field name.   Insert/Delete field: Ctrl-N/Ctrl-U

Field names begin with a letter and may contain letters, digits and underscores

شكل (٣ - ١)

```
CLEAR
SET  TALK OFF
USE  TEST  ORDER  NUM
DO WHILE  .T.
CLEAR
DECLARE  MT [4]
MT[1] = 0
MT[2] = SPACE  (10)
MT[3] = SPACE  (12)
MT[4] = 0
@2,2 to 18,77 DOUBLE
@3,28 to 5,60
@4,31 say "ESSENTIAL DATA"
@7,5 say "NUMBER   :" GET MT[1] PICTURE  "99"
@9,5 say "NAME        :"GET MT[2] FUNCTION "!"
@11,5 say "ADDRESS :" GET MT[3]
@13,5 SAY "TELEPHONE:" GET MT[4] PICTURE   ;
  "9999"
READ
DEC = " "
@15,10 SAY "IS THIS RIGHT?  Y/N  OR X  TO EXIT";
  GET DEC FUNCTION "!" VALID DEC$"YNX"
READ
IF DEC$"X"
EXIT
ENDIF
IF DEC$"N"
LOOP
ENDIF
APPEND FROM ARRAY MT
ENDDO
RETURN
```

١- فى السطر (٦) يتم إعلان المصفوفة الأحادية MT والتى تتكون من أربعة عناصر .

٢- من السطر (٧) الى السطر (١٠) يتم تخصيص عناصر المصفوفة لنوع البيانات التى يدخلها المستخدم .

٣- فى سطر (٢٨) ، وبعد التأكد من صحة البيانات ، يتم نسخ البيانات المخزنة داخل المصفوفة MT الى سجل واحد فى نهاية ملف قاعدة البيانات (TEST. dbf) .

عند تنفيذ البرنامج يتم عرض النموذج شكل (٣ - ٢) ، حيث تُستقبل البيانات التى يدخلها المستخدم .

Caps

```
ESSENTIAL DATA

NUMBER     : 0
NMAE       :
ADDRESS    :
TELEPHONE  : 0
IS THIS RIGHT ?  Y/N  OR  X  TO  EXIT
```

شكل (٣ - ٢)

الأمر Public

يتيح لك التعامل مع الحقول المتغيرة (memvars) التى أنشئت فى أى برنامج فرعى ، وأن تتعامل معها من خلال البرنامج الرئيسى او برنامج فرعى آخر .

الصورة العامة :

PUBLIC <memvar list >/[ARRAY <array element list >]

١- حقول الذاكرة المتغيرة (Memvars) التى تريد إعلانها عامة (PUBLIC) يجب أن تعلن قبل إعطائها أى قيم .

٢- تحديد حقول الذاكرة من نقطة الأوامر (Dot prompt) يعنى ذلك إعلانها عامة (Public).

٣- قيم حقول الذاكرة المعلنة عامة قد تختفى مؤقتا عند استدعاء برنامج فرعى أو ملف إجراءات أو عند إعلان حقول متغيرة لها نفس الإسم بالإعلان "خاص" (PRIVATE) .

٤- الحقول المتغيرة التى حفظت (Saved) داخل ملف على القرص المغنط (Disk) ثم استدعيت للذاكرة (Restored) يتم إعلانهم عامة إذا كان الإستدعاء من خلال نقطة الأوامر ويتم إعلانهم خاص إذا كان الإستدعاء من خلال برنامج .

ملحوظة : المصفوفة (Array) تكون عامة (Public) إذا كتب الأمر (Declare) من خلال نقطة الأوامر(Dot prompt) ، وتكون خاصة (Private) إذا كتب الأمر (Declare) من خلال برنامج ...

```
* Main program
Clear
Do while .T.
Do init-vars
   .
   .
   .
ENDDO
Return
PROCEDURE  init-vars
PUBLIC mname,  mnumb,  mjob
   mname = space (20)
   mnumb = 0
   mjob = space (10)
   Return
```

لاحظ أن الحقول المتغيرة (Mname) و (Mnumb) و (Mjob) التى أنشئت وتم إعلانها عامة من خلال الإجراء (init-vars) سوف يتم تنفيذها خلال البرنامج الرئيسى (Main program) ، أى أنه يمكن التعامل مع حقول متغيرة أنشئت فى المستوى الأدنى ، بشرط أن تعلن هذه الحقول (Public).

إظهار حقول الذاكرة

باستخدام ??/? :

مثال :

```
.Store "MAHA" to mname
.? mname
.MAHA
```

إظهار محتويات الذاكرة

باستخدام الأمر List/Display memory

مثال :

```
.STORE "MAHA" TO NAME
MAHA
.STORE "CAIRO" TO CITY
CAIRO
.DISPLAY MEMORY
```

User Memory Variables

```
CITY     pub  C  "CAIRO"
NAME     pub  C  "MAHA"
```

z out of 500 memvars defined (and 0 array elements)

User MEMVAR/RTSYM Memory Usage

```
 800 bytes used for 1 membar blocks (max=10)
 850 bytes used for 1 rtsym blocks (max=10)
   0 bytes used for 0 array element memvars
   9 bytes used for 2 memvar character strings
3659 bytes  total
```

Print System Memory Variables

```
 BOX       pub  L  .T.
-TABS      pub  C  ""
-PCOLNO    pub  N        21  (21.00000000)
-PLINENO   pub  N        24  (42.00000000)
```

حفظ حقول الذاكرة المتغيرة على القرص الممغنط DISKETTE

محتويات جميع أنواع حقول الذاكرة المتغيرة تضيع بمجرد الخروج من dBase IV او عند إطفاء الحاسب ، ولنفترض أنك تريد الإحتفاظ بهذه المحتويات لاستخدامها مرة أخرى .. يمكنك ذلك باستخدام الأمر SAVE TO لحفظ جميع حقول الذاكرة أو بعضها داخل ملف على القرص الممغنط .

الشكل العام :

SAVE TO <filename >[ALL like/except <skeleton >]

مثال -١ : Save to file1

يحفظ جميع حقول الذاكرة المتغيرة داخل الملف (File1) .

مثال -٢ : Save all like A* to file2

يحفظ جميع الحقول التى تبدأ بالحرف A داخل الملف (File2) .

مثال -٣ : Save all except ??B* to file3

يحفظ جميع الحقول داخل الملف (File3) فيما عدا الحقول التى يكون ثالث حرف فى إسمها الحرف B .

حذف محتويات الذاكرة

يتم حذف محتويات الذاكرة كلها أو بعضها باستخدام الأمر (Release) .

الإستخدام العام :

1- Release <memvar list>
2- Release all [like /except <skeleton >]

مثال :

فى المثال التالى يتم أولا حفظ حقول الذاكرة داخل ملف إسمه (MFile) ،
ثم يتم حذف جميع حقول الذاكرة المتغيرة باستخدام الأمر (Release all)

```
.SAVE TO MFILE
.RELEASE ALL
.DISPLAY MEMORY
                                                       Caps
                    User Memory Variables

0  out of 500  memvars defined  (and 0 array elements )

              User MEMVAR/RTSYM Memory Usage

  0 bytes used for 0 memvar blocks (max= 10)
850 bytes used for 1 rtsym blocks (max= 10)
  0 bytes used for 0 array element memvars
  0 bytes used for 0 memvar character strings
850 bytes total

                 Print System Memory Variables

- BOX          pub  L  .T.
- TABS         pub  C  ""
- PCOLNO       pub  N           21  (21.000000)
- PLINENO      pub  N           47  (47.000000)
- PAGENO       pub  N           11  (11.000000)
- ALIGNMENT    pub  C  "LEFT"
Press any key to continue ..
```

لاحظ أن الأمر Release قد حذف جميع محتويات الذاكرة - فيما عدا حقول ذاكرة النظام (System Variables) طبعا .

إستعادة حقول الذاكرة من القرص الممغنط

ولاستعادة حقول الذاكرة التى تم تخزينها داخل الملف (mfile) الى الذاكرة مرة أخرى ، إستخدم الأمر : Restore From mfile .

مثال :

```
.RESTORE FROM MFILE
.DISPLAY MEMORY

                    User Memory Variables

CITY     pub  C  "CAIRO"
NAME     pub  C  "MAHA"
2 out of 500  memvars  defined (and 0 array elements)

              User MEMVAR/RTSYM Memory Usage

2800 bytes used for 1 memvar blocks (max= 10)
 850 bytes used for 1 rtsym blocks (max= 10)
   0 bytes used for 0 array element memvars
   9 bytes used for 2 memvar character strings
3659 bytes total

                 Print System Memory Variables
- BOX        pub  L  .T.
- TABS       pub  C  ""
- PCOLNO     pub  N            21  (21.000000)
- PLINENO    pub  N            53  (53.000000)
Press any key to continue ..
```

لاحظ وجود حقل الذاكرة المتغيرة (CITY) والحقل (NAME) بالذاكرة مرة أخرى نتيجة لاستخدام الأمر Restore.

الفصل الرابع

تراكيب التحكم

* أنواع تراكيب التحكم.
* الحلقة الشرطية IF ... ENDIF -
تداخل NESTING الحلقة الشرطية.
* الوظيفة ( ) IIF.
* الأمر DO CASE ... ENDCASE.
* الدوران.
* الحلقة الدوارة المشروطة.
* الحلقة الدوارة اللانهائية.
* حلقة المسح الدوارة.
* التداخل.

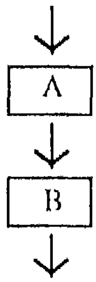
الفصل الرابع

تراكيب التحكم (CONTROL STRUCTURES)

تراكيب التحكم هى الأدوات الأساسية التى تحدد التتابع الذى تُنفذ به الأوامر (Commands) او الجزيئيات البرمجية (Modules) ، وهى :

١- التتابع (SEQUENCE)

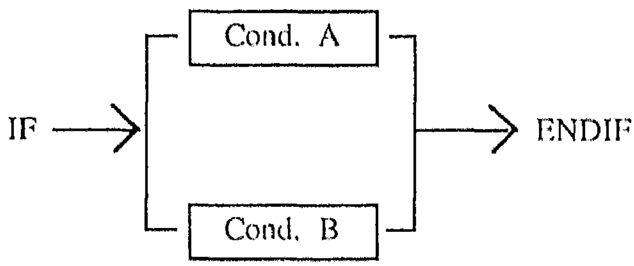
والمقصود به هو أن تنفيذ الأمر B يجب أن يتبع تنفيذ الأمر A .



Sequence

٢- الإختيار (SELECTION)

إختيار فعل من مجموعة من الإختيارات يعتمد على الشرط المبين ؛ (IF .. ELSE .. ENDIF) .




```
IF condition
    DO TASK-1
ELSE
    DO TASK-2
ENDIF
```

لاحظ أن التركيبة لها مدخل واحد (IF) ومخرج واحد (ENDIF) .

٣- الإختيار المتعدد MULTIPLE SELECTION

وهو اختيار تنفيذ فرع من عدة فروع ؛ (DO case .. END case) .

```
  DO CASE
Case-1
    do task-1
Case-2
    do task-2
Case-3
    do  task-3
 END CASE
```

٤- التكرار REPETITION

وهو أن يقوم البرنامج بتكرار تنفيذ نفس الخطوات مادام الشرط المحدد صحيح ؛ (Do while .. ENDDOD)
(SCAN .. ENDSCAN)

الحلقة الشرطية IF .. ENDIF

يقوم بتنفيذ مجموعة من الأوامر فى حالة تحقق الشرط .
الصورة العامة :

```
IF <condition >
   <commands >
[ELSE
   <commands >]
ENDIF
```

وإذا لم يتحقق الشرط ، فإما أن ينفذ مجموعة أخرى من الأوامر أو لا ينفذ شيئا على الإطلاق ، ويتم الإنتقال للأمر الذى يلى ENDIF .

مثال -١ :

```
IF  salary > 1000
Replace tax with salary * 0.02
ENDIF
```

فى حالة أن يكون الراتب (Salary) أكبر من ١٠٠٠ ، فإن قيمة الضريبة (Tax) = المرتب * ٠,٠٢

مثال -٢ :

```
IF  salary > 1000
    Replace tax with salary * 0.02
ELSE
    Replace tax with salary * 0.01
ENDIF
```

فى حالة أن يكون المرتب أكبر من ١٠٠٠ ، فإن الضريبة = قيمة المرتب * ٠,٠٢ ؛
فإن لم يتحقق هذا الشرط ، أى أن يكون المرتب = أو أقل من ١٠٠٠ ، فإن الضريبة = قيمته * ٠,٠١ .
وبعد تنفيذ الأمر ، أى حساب قيمة الضريبة ، فإنه يتم الإنتقال الى الأمر الذى يلى ENDIF لتنفيذه .

التداخل NESTING

يستخدم فى حالة وجود أكثر من اختيارين ؛ IF .. ENDIF

مثال -٣ :

```
1  IF salary > 1000
2  replace tax with salary * 0.02
3  ELSE
4  IF salary > 750
5  replace tax with salary * 0.01
6  ELSE
7  IF salary > 500
8  replace tax with salary * 0.008
9  ENDIF
10 ENDIF
11 ENDIF
```

١- السطر (١)
فى حالة أن يكون المرتب (Salary) أكبر من ١٠٠٠
٢- السطر (٢)
إجعل الضريبة (Tax) = المرتب * ٠,٠٢

٣- السطر (٣)

إذا لم يتحقق الشرط ، أى إن لم يكن المرتب = أو أقل من ١٠٠٠

٤- السطر (٤)

فى حالة أن يكون المرتب أقل من أو يساوى ١٠٠٠ ، وأكبر من ٧٥٠

٥- السطر (٥)

إجعل الضريبة = المرتب * ٠,٠١

٦- السطر (٦)

فى حالة عدم تحقق شرط أن يكون المرتب أكبر من ٧٥٠

٧- السطر (٧)

فى حالة أن يكون المرتب أقل من أو يساوى ١٠٠٠ ، وأكبر من ٥٠٠

٨- السطر (٨)

إجعل الضريبة = المرتب * ٠,٠٠٨

٩- السطر (٩)

تغلق الحلقة الشرطية التى تبدأ بالأمر IF salary > 500 فى السطر ٧ .

١٠- السطر (١٠)

تغلق الحلقة الشرطية التى تبدأ بالأمر IF salary > 750 فى السطر (٤).

١١- السطر (١١)

تغلق الحلقة الشرطية التى تبدأ بالأمر IF salary >1000 فى السطر (١).

فى المثال السابق ، يلاحظ الآتى :

١- الحلقة الشرطية التى تبدأ بالأمر IF salary >750 فى السطر (٤) وتنتهى بالأمر ENDIF فى السطر (١٠) مدمجة تماما أو متداخلة (Nested) - داخل الحلقة الشرطية التى تبدأ من السطر (١) وتنتهى فى السطر (١١).

٢- الحلقة الشرطية التى تبدأ بالأمر IF salary >500 فى السطر (٧) وتنتهى بالأمر ENDIF فى السطر (٩) متداخلة تماما ، أى أنها تبدأ بالأمر IF وتنتهى بالأمر ENDIF داخل الحلقة الشرطية التى تبدأ بالأمر IF فى السطر (٤) وتنتهى بالأمر ENDIF فى السطر (١٠).

الوظيفة IIF ( )

IIF (<condition >, <expl >, <exp2 >)

أ – فى حالة تحقق الشرط فإنها تعطى التعبير الأول (Exp1) .

ب – فى حالة عدم تحقق الشرط ، فإنها تعطى التعبير الثانى (Exp2) .

مثال :

```
IF salary > 1000
replace tax with salary * 0.02
ELSE
replace tax with salary * 0.01
ENDIF
```

باستخدام IIF ( ) تستطيع دمج السطور الخمسة السابقة فى سطر واحد كما يلى :

```
Replace tax with IIF (salary >1000, salary * 0.02, salary * 0.01)
```

١– فى حالة تحقق الشرط (salary > 1000) ، فإن الوظيفة IIF () تعطى التعبير الأول وهو (salary * 0.02) ، وبذلك يصبح سطر الأمر كاملا كما يلى :

. replace tax with salary * 0.02

٢– فى حالة عدم تحقق الشرط (salary > 1000) فإن الوظيفة IIF () تعطى التعبير الثانى وهو (salary * 0.01) وبذلك يصبح سطر الأمر كاملا كما يلى :

. replace tax with salary * 0.01

الأمر DO CASE .. ENDCASE

يقوم بتنفيذ اختيار من مجموعة اختيارات ، ويفضل عن الأمر (IF...ENDIF) فى حالة وجود أكثر من اختيارين.

مثال :

```
Do while .T.
Clear
? "                    Main Menue "
?
? "    1 = Task-1"
? "    2 = Task-2"
? "    3 = Task-3"
? "    4 = Task-4"
? "    5 = EXIT"
?
Wait "Enter your choice (5 to exit)" to mchoice
Do case
case mchoice = '1'
do task-1
case mchoice = '2'
do task-2
case mchoice = '3'
do task-3
case mchoice = '4'
do task-4
case mchoice = '5'
EXIT
Otherwise
? "please choice from 1 to 4  (5 to exit)"
Wait
Loop
ENDCASE
ENDDO
Return
```

من المثال السابق نلاحظ التالى :

١- فى حالة قيام المستخدم باختيار أحد خيارات القائمة (من ١ الى ٤) يتم التفريع (Branching) لتنفيذ برنامج فرعى (Subprogram) أو إجراء (Procedure) .

٢- بعد تنفيذ البرنامج الفرعى أو الإجراء ، يتم الإنتقال الى الأمر الذى يلى ENDCASE وهو الأمر ENDDO الذى يسبب الإنتقال الى الأمر Clear (السطر الثانى فى البرنامج) ليبدأ عرض القائمة مرة أخرى .

٣- فى حالة قيام المستخدم باختيار (٥) للخروج ، فإنه سوف يتم تنفيذ الأمر EXIT الذى يسبب الخروج من الحلقة الدوارة - .. (Do while .T.) (ENDDO) فيتم الإنتقال الى الأمر التالى للأمر ENDDO ، وهو الأمر Return الذى يسبب الإنتقال الى الموقع الذى نفذ منه هذا البرنامج ، وهو إما البرنامج المستدعى (Calling program) او نقطة الأوامر (Dot prompt) أو مركز التحكم (Contorl center) .

٤- فى حالة قيام المستخدم بضغط أى مفتاح آخر غير المتاح فى القائمة (من ١ الى ٥) ، فإنه يتم الإنتقال الى الأمر الذى يلى الأمر (Otherwise) وهو السطر الذى يعرض رسالة على الشاشة لحث المستخدم على اختيار رقم متاح فى القائمة ، ثم يتم تنفيذ الأمر التالى وهو الأمر (Wait) الذى يسبب إيقاف الشاشة الحالية مع عرض رسالة press any key to continue . ومتى ما قام المستخدم بضغط أى مفتاح فإنه يتم الإنتقال الى الأمر التالى وهو الأمر (Loop) الذى يسبب الإنتقال الى الأمر (Clear) فى السطر الثانى فى البرنامج ليبدأ عرض القائمة مرة أخرى .

الدوران LOOPING

الحلقة الدوّارة (Do while .. ENDDO) ، تقوم بتكرار الأوامر الواقعة بين (Do while) و (ENDDO) مادام الشرط المحدد صحيحا .

الصورة العامة :

```
Do while <condition >
<commands >
[loop]
[exit]
ENDDO
```

١- الأمر Loop يمنع تنفيذ الأوامر التى تأتى بعده ، ويتم الإنتقال الى بداية الحلقة الدوارة .

ملحوظة : يستخدم الأمر Loop فقط تحت شرط ، أى أنه يستخدم مع الأمر (IF) أو الأمر (Do case) متداخلة (Nested) داخل الحلقة الدوارة (Do while) .

٢- الأمر Exit يؤدى الى الخروج من الحلقة الدوارة ، ويبدأ التنفيذ للأمر التالى لنهاية الحلقة الدوارة (ENDDO) مباشرة .

الحلقة الدوارة المشروطة

```
DO  while  < condition >
.
.
ENDDO
```

يتم تكرار الأوامر الواقعة بين طرفيها مادام الشرط المحدد صحيحاً .

مثال :

```
X = 1
Do while X  < 200
X = X + 1
ENDDO
```

فى هذا المثال يتم إتاحة فترة تأخير لإعطاء المستخدم الفرصة لقراءة رسالة معينة على الشاشة . وفيه يتم الدوران فيما بين السطرين :
(Do while X < 200) و (ENDDO) مادامت قيمة X أقل من ٢٠٠ ، ومع كل دورة تزداد قيمة X بمقدار واحد صحيح ، أى أنه يتم الدوران بمقدار ٢٠٠ مرة.
وبمجرد عدم تحقق الشرط ، أى أن تكون قيمة X = ٢٠٠ ، يتم الإنتقال للأمر الذى يلى سطر (ENDDO) .

الحلقة الدوارة المشروطة

Do while .NOT. EOF ( ) … ENDDO

فيها يتم تكرار الأوامر الواقعة بين … Do while .NOT. EOF ( ) وENDDO مادام مؤشر السجلات (Records pointer) داخل الملف ، فإذا انكسر الشرط ، أى لم يعد مؤشر السجلات داخل الملف ، فإنه يتم الإنتقال الى الأمر التالى لـ ENDDO لتنفيذه .

مثال :

```
1 Use store
2 Do while .NOT.  EOF ()
3 clear
4 list next 8 itemno,  itemname,  balance
5 skip
6 ENDDO
```

من المثال السابق نلاحظ التالى :

١- السطر (٣)
الأمر Clear لمسح الشاشة .

٢- السطر (٤)
لعرض الحقول (Itemno, Itemname, Balance) - لعدد ٨ سجلات تالية

٣- السطر (٥)
الأمر Skip للإنتقال الى السجل رقم (٩) فى الملف Store .

٤- السطر (٦)

عند الوصول الى نهاية الحلقة الدوارة (ENDDO) فإنه يتم تنفيذ إحدى الحالتين التاليتين :

أ - حالة عدم تحقق الشرط المحدد ، وهو خروج مؤشر السجلات (Pointer) من الملف ، وفى هذه الحالة يتم الخروج من الحلقة الدوارة ويتم الإنتقال الى الأمر التالى لنهايتها .

ب - حالة استمرار تحقق الشرط المحدد وهو استمرار وجود مؤشر السجلات داخل الملف Store وفى هذه الحالة يتم الإنتقال الى الأمر Clear فى السطر (٣) لمسح الشاشة وعرض ثمانية سجلات أخرى على الشاشة.

الحلقة الدوارة اللانهائية Do while .T. .. ENDDO

يتم تكرار الأوامر الواقعة بين بداية الحلقة الدوارة ونهايتها عددا لا نهائياً من المرات ، ولذلك فهى تستخدم فى البرامج بتحفظ ، ويجب أن يدمج الأمر Exit داخل الحلقة الدوارة اللانهائية ليتم تنفيذه عند تحقق شرط معين تقوم بتحديده داخل الحلقة الدوارة اللانهائية .

```
Do while  .T.
Clear
? "                 Main Menue "
?
? "    1 = Task-1"
? "    2 = Task-2"
? "    3 = Task-3"
? "    4 = Task-4"
? "    5 = EXIT"
?
Wait "Enter your choice (5 to exit)" to mchoice
Do case
case mchoice = '1'
do task-1
case mchoice = '2'
do task-2
case mchoice = '3'
```

```
do task-3
case mchoice = '4'
do task-4
case mchoice = '5'
EXIT
Otherwise
? "please choice from 1 to 4  (5 to exit)"
Wait
Loop
ENDCASE
ENDDO
Return
```

حلقة المسح الدوارة SCAN ... ENDSCAN

تختار سجلات (Records) معينة من ملف قاعدة البيانات ثم تنفذ أوامر معينة عليها وتعتبر مرادف مبسط للحلقة الدوارة :

```
Do while
.
.
ENDDO
```

الصورة العامة:

```
SCAN[<scope>][FOR<condition>][WHILE<condition>]
[<commands>]
[loop]
[EXIT]
ENDSCAN
```

١- الأمر Loop يسبب الإنتقال لبداية الحلقة الدوارة (SCAN).
٢- الأمر EXIT ينهى العمل باستخدام SCAN ، ويتم الإنتقال للأمر الذى يلى سطر EXIT مباشرة.
٣- FOR <cond> يحدد سجلات (Records) الملف التى سوف يتم التعامل معها.
٤- WHILE <cond> تحديد إضافى لسجلات الملف التى سوف يتم التعامل معها ، ويكرر التنفيذ مادام الشرط متحقق.

٥- SCAN يبدأ تنفيذ المسح من بداية الملف أو من أول سجل فى الملف المفهرس (Inedexed) إذا لم تحدد له بداية أخرى من خلال استخدام While <cond > أو FOR <cond > .

٦- SCAN تكرر التنفيذ حتى نهاية الملف أو آخر سجل (Record) فى الملف المفهرس ما لم تحدد له نهاية أخرى من خلال استخدام FOR <cond > أو WHILE <cond > أو EXIT .

٧- مؤشر السجلات (Pointer) يظل فى نفس السجل الذى كان فيه عندما تم إنهاء التنفيذ باستخدام SCAN .

مثال :

```
1 Use master
2 SCAN FOR name = "AHMED" while salary > 200
3 ? name, salary, depart.
4 ENDSCAN
```

من المثال السابق نلاحظ التالى :

١- السطر (٢)

يقوم بتحديد شروط لاختيار سجلات محددة من الملف (Master) تحقق الشرط name = "AHMED" ، ويكون فيه الراتب أكبر من ٢٠٠.

٢- السطر (٣)

لعرض حقول .. Name ,Salary ,Dept. .

٣- السطر (٤)

يتم تنفيذ إحدى حالتين :

أ – فى حالة وصول مؤشر السجلات لنهاية الملف ، فإنه يتم الخروج من حلقة المسح الدوارة ، ويتم الإنتقال للأمر التالي لنهاية حلقة المسح (ENDSCAN) .

ب – فى حالة استمرار وجود مؤشر السجلات داخل الملف فيتم الإنتقال الى السطر (٣) حيث يتم عرض الحقول المبينة والتى تحقق الشرط المحدد فى السطر (٢) .

NESTING التداخل

كل تركيبة من تراكيب التحكم السابق شرحها يمكن تداخلها مع تركيبة أخرى تداخلا تاماً .. هكذا :

```
Do while .T.
Do while .NOT. EOF ( )
IF <cond>
.
.
.
ELSE
Do case
case 1
.
.
END CASE
  ENDIF
ENDDO
ENDDO
```

مثال :

```
Use store ORDER itemno
Do while .T.
Loop
Clear
Enter "please enter the itemno (0 to exit)" To
MNO
Seek MNO
IF EOF ( )
?
? "Please - re-enter the itemno"
Wait
ENDIF
IF MNO = 0
EXIT
ENDIF
ENDDO
RETURN
```

الفصل الخامس

إنشاء القوائم

* تعريف القائمة.
* إنشاء القائمة الأفقية.
* إنشاء القائمة الرأسية.
* تنفيذ القوائم.
* مسح القوائم.

الفصل الخامس

إنشاء القوائم MENUS

تعريف القائمة :

هى مجموعة ثابتة من الإختيارات تتيح للمستخدم اختيار أحداها ، وتعطى dBase IV إمكانية إنشاء ثلاثة أنواع من القوائم :

١- Bar Menu

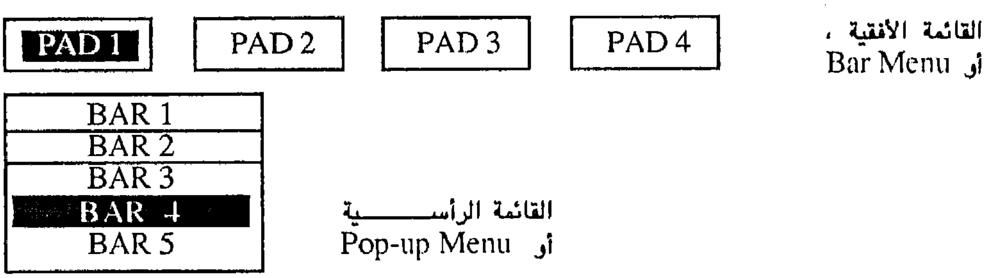
وهى قائمة أفقية ، عبارة عن شريط أو أكثر بأعلى الشاشة .

٢- Pop-up Menu

وهى قائمة إنبثاقية رأسية تعرض عندما يقوم المستخدم باختيار أحد الخيارات من قائمة أفقية أو من إنبثاقية أخرى ، وتستطيع تحديد مكان ظهورها فى أى مكان على الشاشة .

٣- Pull-down Menu

وهى عبارة عن قائمة إنبثاقية متصلة بالقائمة الشريطية ومن أحد خياراتها . شكل (٥ – ١) .



شكل (٥ – ١)

ملحوظة: الفرق بين قائمة Pop-up وقائمة Pull-down هو فى أنه يجب الضغط على مفتاح <enter> لإظهار القائمة Pop-up على العكس من الأخرى التى تكون متصلة بعنصر القائمة الأفقية.

أولا : إنشاء القائمة الأفقية (Bar Menu)

١- يتم تعريفها بالأمر .. Define Menu <menu name >

٢- تعرف إختيارات القائمة بالأمر ..

Define PAD <m> OF <bar menu> PROMPT <prompt string >
AT <co-ords > MESSAGE <message string >

Prompt الجملة التى تعرض داخل اختيار القائمة (PAD) .

Message رسالة تكون مرتبطة مع الإختيار (PAD) وتكون بطول حتى ٧٩ حرفاً ، وتظهر أسفل منتصف الشاشة عندما يكون المؤشر على (PAD) مرتبطة به .

٣- لتنفيذ فعل ما ، كرد فعل على قيام المستخدم باختيار أحد اختيارات القائمة ..

ON Selection PAD <pad name > OF <menu name >
[Ativate POP-UP <popup menu >]

حيث يقوم كل (Pad) بعرض قائمة Pop-up تحتوى على اختيارات

تنفيذ القوائم

أولاً : القائمة الأفقية (Bar Menu)

باستخدام الأمر ..

Activate MENU <menu name > [PAD <pad name >]

الذى يعرض القائمة ويجعلها قابلة للتعامل مع الإختيارات .

ثانيا : تنفيذ القائمة الرأسية (Pop-up)

باستخدام الأمر ..

Activate POPUP <popup name >

الذى يعرض القائمة التى سبق تعريفها ويجعلها قابلة للتعامل مع الإختيارات.

مسح القوائم

أ - مسح قائمة واحدة من الشاشة ؛ باستخدام الأمر Deactivate Menu .

ب - مسح من الذاكرة ؛ باستخدام الأمر Release Menu <menu name >.

جـ - مسح جميع القوائم ؛ باستخدام الأمر Clear Menus .

مثال :

لإنشاء قائمة أفقية تحتوى على الإختيارات : Update, Position, Exit ، وقوائم رأسية لكل منها كالتالى :

قائمة Update تحتوى على : Append, Edit, Display, Delete .

قائمة Position تحتوى على : Go To, Skip .

قائمة Exit تحتوى على : EXIT TO dot prompt, EXIT TO DOS .

١- يجب تعريف القائمة الأفقية (Bar Menu) ..

Define Menu M

٢- نعرّف كل اختيار بالقائمة الأفقية ؛ هكذا :

```
Define PAD update of M prompt "update" AT 2,4
Define PAD position of M prompt "position" AT 2,22
Define PAD EXIT of M prompt "EXIT" AT 2,40
```

٣- نعرّف القائمة الرأسية لكل اختيار ، وكذلك كل اختيار فى القوائم الرأسية :

```
Define popup upd-pop from 4,4 To 8,14
Define bar 1 of upd-pop prompt "APPEND"
Define bar 2 of upd-pop prompt "EDIT"
Define bar 3 of upd-pop prompt "DISPLAY"
Define bar 4 of upd-pop prompt "Delete"
Define popup pos-pop from 4,22 TO 7,30
Define Bar1 of pos-pop prompt "SKIP"
Define Bar2 of pos-pop prompt "Go To"
Define popup exit-pop from 4,40 To 7,50
Define Bar1 of exit-pop prompt "EXIT TO DOS"
Define Bar2 of exit-pop prompt "EXIT to dot prompt"
```

٤- لتنفيذ الفعل المترتب على اختيار المستخدم لأحد الإختيارات الأفقية ..

```
on pad update of M Activate popup upd-pop
on pad position of M Activate popup upd-pop
on pad EXIT of M Activate popup upd-pop
```

٥- لتنفيذ الفعل المترتب على الإختيار الرأسى ..

```
on selection popup upd-pop Do prog 1
on selection popup pos-pop Do prog 2
on selection popup exit-pop Do prog 3
```

٦- لتنفيذ القوائم ..

```
Activate Menu M PAD update
Return
```

مثال :

لإنشاء قائمة أفقية (Bar Menu) تتكون من ستة عناصر ، هى :
بيانات - أرصدة - بحث - تقارير - صيانة - خروج .

أولاً : أ - إبدأ بتعريف القائمة الأفقية (Bar Menu) ، ثم عرّف اختيارات القائمة .

ب - عرّف القوائم الرأسية (pop-up menu) المرتبطة باختيارات القائمة الأفقية.

```
PROCEDURE MDEF
Define Menu MINV
Define PAD U OF MINV prompt "بيانات" ;
        at 3,3; mess " enter إختر باستخدام الأسهم ، ثم اضغط"
Define PAD E OF MINV prompt "أرصدة" ;
        at 3,15; mess "تعديل أرصدة الأصناف الإستراتيجى"
Define PAD S OF MINV prompt "بحث" ;
        at 3,27; mess "البحث عن أرصدة حالية وسابقة"
Define PAD R OF MINV prompt "تقارير" ;
        at 3,40; mess "تقارير إجمالى الأرصدة وتقرير ح"
Define M OF MINV prompt "صيانة" ;
        at 3,53; mess "نسخ أرصدة للرجوع إليها"
Define PAD X OF MINV prompt "خروج" ;
        at 3,66; mess "خروج من البرنامج"

*    POPUP UP
DEFINE POPUP PU FROM 8,3 TO 16,25
DEFINE BAR 1  OF PU PROMPT  "تعديل بيانات"  skip
DEFINE BAR 2  OF PU PROMPT  "==============="  SKIP
DEFINE BAR 3  OF PU PROMPT  "إضافة صنف"
DEFINE BAR 4  OF PU PROMPT  "شطب صنف"
DEFINE BAR 5  OF PU PROMPT  "تعديل صنف"
DEFINE BAR 6  OF PU PROMPT  "خروج الى القائمة"
```

ثانيا : لتنفيذ الأفعال المقابلة لاختيارات القائمة الأفقية ..

```
*   SELECTION PAD
ON SELECTION PAD U OF MINV ACTIVATE POPUP PU
ON SELECTION PAD E OF MINV ACTIVATE POPUP PE
ON SELECTION PAD S OF MINV Do SEARCH
ON SELECTION PAD R OF MINV ACTIVATE POPUP PR
ON SELECTION PAD M OF MINV Do PRM
ON SELECTION PAD R OF MINV ACTIVATE POPUP PX
```

ثالثا : لتنفيذ الأفعال المقابلة لاختيارات القائمة الرأسية (Pop-up)

```
*   SELECTION POPUP
ON SELECTION POPUP PU DO PRU
ON SELECTION POPUP PU DO PRE
ON SELECTION POPUP PU DO PRR
ON SELECTION POPUP PU DO PRX
```

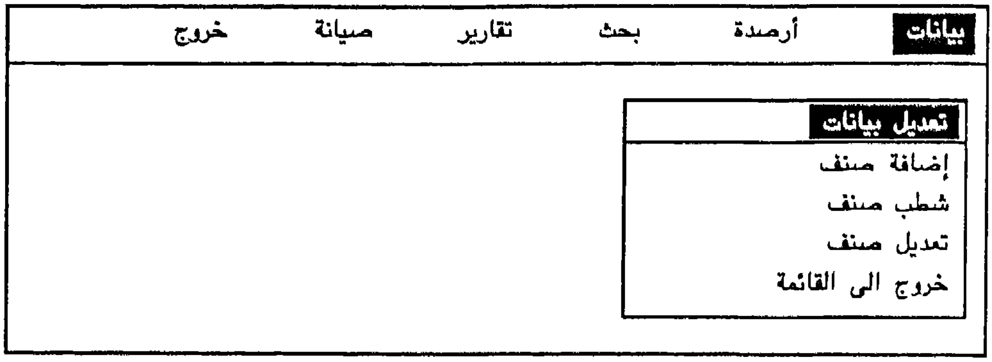
الإجراء PRU :

```
PROCEDURE PRU
DO CASE
CASE BAR ( ) = 3
DO RPA
CASE BAR ( ) = 4
DO DEL
CASE BAR ( ) = 5
DO ED
CASE BAR ( ) = 6
clear
RETURN
ENDCASE
RETURN
```

رابعاً : لتنفيذ القائمة ..

```
ACTIVATE MENU MINV PAD U
```

ونتيجة لتنفيذ القائمة تظهر على الشاشة القائمة المبينة فى شكل (٥ – ٢) .



إختر باستخدام الأسهم ثم اضغط .. enter ..

شكل (٥ – ٢)

الفصل السادس

الجزيئات المبرمجة

* تعريف الجزيئات المبرمجة.
* ملف الإجراءات.
* طريقة إنشاء ملف الإجراءات.
* التعامل مع ملف الإجراءات من داخل البرنامج.
* إستخدام المعطيات مع الإجراءات.
* المعطيات.
* وظائف المستخدم.

الفصل السادس

الجزئيات المبرمجة PROGRAM MODULES

يتكون البرنامج فى البرمجة التركيبية (Modular programming) من برنامج رئيسى ، ومجموعة من الجزئيات المبرمجة (Modules) .

الجزئيات المبرمجة من الممكن أن تكون :

١- برامج فرعية (Sub programs) .
٢- إجراءات (Procedures) .
٣- معطيات (Parameters) .
٤- وظائف المستخدم (UDF).

يُنفذ البرنامج من نقطة الأوامر (Dot prompt) بكتابة الأمر التالى :
Do <filename >
ومن الممكن تنفيذ برنامج فرعى أو إجراء أو معطيات من خلال البرنامج الرئيسى باستخدام الأمر .. Do <filename > ، فيتم الانتقال للجزيئ المبرمج ويتم تنفيذه حتى آخر أمر فيه ، وعندما يصل الى الأمر (Return) يتم الإنتقال مرة ثانية الى البرنامج الرئيسى ويبدأ تنفيذ الأمر التالى .

مثال لبرنامج فرعى :

```
SET TALK OFF
CLEAR
DO MAIN
ACTIVATE POPUP TEST-POP
ON SELECTION POPUP TEST-POP DO PROG 1
CLEAR ALL
SET TALK ON
RETURN
```

مثال لإجراء :

```
PROCEDURE PROG 1
DO CASE
CASE BAR ( ) = 1
DO SUBPROG 1
CASE BAR ( ) = 2
DO SUBPROG 2
CASE BAR ( ) = 3
DO SUBPROG 3
END CASE
RETURN
```

ملف الإجراءات PROCEDURES FILE

ملف الإجراءات هو ملف يحتوى على مجموعة من الجزئيات البرمجية (Modules) تسمى (Procedures) أى إجراءات ، يصل عددها الى ١١٧. إجراءً فى الملف الواحد كحد أقصى ، أو حسب ما تسمح به ذاكرة الحاسب الرئيسية (RAM) .
والإجراءات تتيح تقليل إشغال القرص الممغنط (Disk) ، وزيادة سرعة تنفيذ البرنامج ، نظرا لأنها توضع داخل ذاكرة الحاسب الرئيسية .

طريقة إنشاء ملف الإجراءات

يُنشأ ملف الإجراءات بنفس طريقة إنشاء ملف البرنامج (Program file) باستخدام الأمر .. Modify command <filename > .
وفى بداية كل إجراء يكتب Procedure ، وفى نهايته يكتب Return .

مثال :

```
*  PROCEDURE FILE .. TEST PRG
PROCEDURE A
<commands >
RETURN
.
.
.
PROCEDURE Z
<commands >
RETURN
```

التعامل مع ملف الإجراءات من داخل البرنامج

يكتب الأمر .. Set procedure to <procedure filename > ، فى بداية البرنامج .

وعند الرغبة فى تنفيذ إجراء معين ، إكتب الأمر .. Do <procedure name > .

مثال :

لتنفيذ الإجراء A من خلال البرنامج الرئيسى (Main prg) ..

```
SET PROCEDURE TO TEST
DO  A
CLOSE PROCEDURE
```

إستخدام المعطيات (PARAMETERS) مع الإجراءات

يتم ذلك بكتابة الأمر Parameter يليه كشف المعطيات بعد الأمر Procedure .

مثال :

```
PROCEDURE
PARAMETER  Y,Z
<command >
RETURN
.
.
RETURN
```

ملحوظة : إذا كان عدد الإجراءات داخل ملف الإجراءات قليلا ، فمن الأفضل وضعهم فى البرنامج الرئيسى (Main program) .

المعطيات

نوع آخر من الجزئيات البرمجية (Modules) يمكن استخدامه كبرنامج منفصل او متحد مع إجراء ما (Procedure) ويمكن إعادة استعماله مع إحداث بعض التغييرات فى مجموعة المعاملات (Parameters list) .
الصورة العامة : Parameters <parameters list > .

مثال :

لنفترض أن المعادلة التى تحكم عملية ما - X = 4 * Y + 70 * Z + 80
يتكرر الرجوع اليها بين الحين والآخر بقيم مختلفة لكل من Y, Z ، فيمكن عمل جزئ مبرمج باستخدام Parameters يحتوى على مجموعة المعاملات (Parameters list) وهى : Y, Z.

```
* EQUATION SOLUTION - EQUAT. PRG
PARAMET  Y, Z
X = 4 * Y + 70 * Z + 80
? X
RETURN
```

وعند تنفيذ الجزء البرمجى بالقيميتن : Y = 5, Z =8 ، إكتب ..

```
DO EQUAT WITH 5,8
660
```

مثال :

لإنشاء إجراء يستخدم المعطيات لكتابة رسالة أسفل الشاشة ويمكن إعادة استخدامه لكتابة رسائل مختلفة ...

```
1 PROCEDURE D-MESSAGE
2 PARAMETERS  D-MESS
3 @ 23,0
4 @ 23, (40-(LEN(D-MESS)/2)) SAY D-MESS
5 RETURN
```

لاحظ أن السطر (٤) لوسطنة الرسالة خلال السطر (٢٣) .   وعند الرغبة فى تنفيذ هذا الإجراء خلال البرنامج لكتابة رسالة معينة مثل :
"Please re-enter ..." ، يكتب الآتى :

Do D-message with "please re-enter..."

وظائف المستخدم USER DEFINED FUNCTIONS

هى وظائف غير موجودة أصلا ضمن وظائف dBase IV ويقوم المستخدم بإنشائها لتقابل احتياجات خاصة به ، مما يعطى مرونة كبيرة جدا فى البرمجة.

وهى عبارة عن إجراء (Procedure) يبدأ بالأمر (Function) ، ويحتوى على أوامر dBase IV، وكشف بالمعطيات (Parameters list) ، وتنتهى بالأمر (Return) الذى يعطى نتيجة الوظيفة ، وتستخدم هذه الإمكانية على وجه الخصوص للتحقق من صحة إدخال أو تعديل البيانات .

كيف تنشئ وظيفة المستخدم ؟

فى برنامج شئون الأفراد يستخدم الكود التالى للتعبير عن مؤهل العامل ...

00, 06, 08, 10, 12, 14, 16, ms, ds

ويستخدم هدا الكود لإدخال البيانات فى حقل المؤهل عند التعيين (R. Grade) وحقل المؤهل حالياً (G. now) .

وعند إدخال الكود فى حقل المؤهل الحالى ، يجب أن تضع فى الإعتبار أن المؤهل الحالى يجب أن يكون مساوياً أو أعلى منه عند التعيين ... فماذا ستفعل ؟
لو كان الكود رقمى لما كانت هناك مشكلة ، إذ أنك فى هذه الحالة يمكنك كتابة التالى :

@... Get Gnow valid Gnow > Rgrade

ولكن فى حالتنا هذه يستخدم كود غير رقمى ، ولا توجد وظيفة أو إمكانية فى dBase IV تتيح عمل تحقق لحظى (Instant validation) للبيانات المدخلة فى حقل (Gnow) عند التعامل مع شاشة إدخال البيانات (الشاشة الكاملة) التى تستخدم الأوامر ...

@... SAY ... GET
READ

وفى هذه الحالة تجد أنك فى أمس الحاجة الى أن تنشئ وظيفة خاصة بك تستخدم مع الإمر VALID للتحقق من صحة إدخال البيانات .

ومن الممكن إنشاء هذه الوظيفة كما يلى :

```
1 FUNCTION CHKGRAD
2 Parameters MGNOW
3 DEC = .F.
4 IF MGNOW$"00-06-07-12-14-16-17-18-MS-DS"
5 DO Case
6 Case RGRAD = 'MS'
7 IF MGNOW = 'MS' .OR. MGNOW = 'DS'
8 Return .NOT. DEC
9 ELSE
10 Return Dec
11 ENDIF
12 Case RGRAD = 'DS'
13 IF MGNOW <>'DS'
14 Return Dec
15 ENDIF
16 ENCASE
17 IF MGNOW <> 'MS' .AND. MGNOW <> 'DS'
18 IF VAL (MGNOW)<VAL (RGRAD)
19 Return Dec
20 ENDIF
21 ENDIF
22 Return .NOT. DEC
23 ELSE
24 Return Dec
25 ENDIF
```

شرح الوظيفة (CHKGRAD)

١- السطر (٣) لإنشاء حقل ذاكرة منطقى يخزن فيه القيمة المنطقية .F.

٢- من السطر (٤) الى (٢٥) حلقة شرطية تعنى أنه فى حالة تحقق الشرط الموضح وهو أن يدخل المستخدم أحد الإختيارات الموضحة فى سطر (٤) ، فإنه يتم تنفيذ الأوامر بين السطر (٤) و (٢٢) ، وفى حالة عدم تحقق الشرط فإنه سيتم تنفيذ سطر (٢٤).

٣- السطر (٨) يعطى نتيجة الوظيفة إيجابية (.T.) فى حالة أن يدخل

المستخدم فى حقل المؤهل الآن (MGNOW) أحد المدخلات MS أو DS ، وأن يكون المؤهل عند التعيين (RGRAD) = 'MS' ، والسطر (١٠) يعطى نتيجة الوظيفة سلبية (.F.) فى حالة عدم تحقق هذا الشرط.

٤- السطر (١٤) يعطى نتيجة الوظيفة سلبية فى حالة أن يكون المؤهل عند التعيين 'DS' ، ويكون حقل المؤهل الآن غير مساويا لحق المؤهل عند التعيين (RGRAD).

٥- السطر (١٩) يعطى نتيجة الوظيفة سلبية (.F.) فى حالة أن تكون القيمة العددية للمؤهل الآن أقل من القيمة العددية لحقل المؤهل عند التعيين.

إستخدام الوظيفة CHKGRAD فى البرنامج

تستخدم فى البرنامج كما يلى :

```
@ 10,3 SAY"GRADE NOW" GET GNOW VALID CHKGRAD (GNOW)
.
.
.
READ
```

وتوجد بعض الأوامر التى لا تستخدم مع وظائف المستخدم (UDF) ، وفي الصفحة التالية كشف بهذه الأوامر .

APPEND APPEND FROM	DEBUG - DEFINE BAR
APPEND FROM ARRAY	DEFINE BOX / MENU
APPEND MEMO ASSIST	DEFINE POPUP / WINDOW
BEGIN / END TRANSACTION	DELETE TAG / DIR
BROWSE CANCEL	EDIT - ERASE
CHANGE CLEAR ALL/FIELD	DELETE FILE - EXPORT
CLOSE ALTERNATE	INDEX - INSERT
CLOSE FORMAT	JOIN - LABEL FORM
CLOSE PROCEDURE	LOAD - LOGOUT
COMPILE	MODIFY COMMAND / FILE
CONVERT COPY	MOVE WINDOW
COPY FILE / INDEX	ON ERROR / ESCAPE / KEY
COPY MEMO / STRUCTURE	ON PAD - ON PAGE
COPY STRUCTURE EXTENDED	ON READERROR
COPY TAG	ON SELECTION PAD
COPY TO ARRAY	ON SELECTION POPUP
CREATE OR MODIFY STRUCTURE	PACK
CREATE FROM	PRINTJOB / ENDPRINTJOB
CREAT VIEW FROM ENVIRONMENT	PROTECT
CREATE / MODIFY APPLICATION	QUIT RE-INDEX
CREATE / MODIFY LABEL	REPORT FORM
CREATE / MODIFY QUERY VIEW	RESTORE
CREATE / MODIFY REPORT	RESTORE MACROS
CREATE / MODIFY SCREEN	RESTORE WINDOW
RESUME	SKIP
ROLLBACK	STEP
SAVE	TRAP
SAVE MACROS	VIEW
SAVE WINDOW	WINDOW
SET SORT	
SUSPEND	
TOTAL	
TYPE	
UPDATE	
ZAP	
CATALOG	
DEBUG	
DEVICE	
FIELDS	
FORMAT	
PROCEDURE	
RELATION	

الأوامر التى لا تُستخدم مع وظائف المستخدِم (UDF)

من خلال تعاملك مع dBase IV وكذلك اطلاعاتك ، تستطيع تكوين مكتبة تحتوى على مجموعة من وظائف المستخدم (UDF) قد تفيدك فى وقت الحاجة اليها . وفيما يلى بعض هذه النماذج التى تستطيع ضمها الى مكتبتك ...

١ الوظيفة (Chkcdouble) تمنع إدخال بيانات سبق إدخالها للملف ، وفى المثال التالى تمنع إدخال رقم صنف فى ملف المخازن سبق إدخاله منعا للتكرار .

```
@ 4,4 SAY "رقم الصنف" GET MNO PICTURE "999";
VALID CHKDOUBLE (MNO)
```

فى السطر السابق يتم استدعاء وظيفة المستخدم (Chkcdouble) للتحقق من عدم تكرار إدخال نفس رقم الصنف ، ويتم البحث عن وظيفة المستخدم تماما كما يتم البحث عن الإجراءات أو البرامج الفرعية .

```
FUNCTION CHKDOUBLE
PARAMETERS NUMBER
DEC = .F.
IF NUMBER = 0
RETURN DEC
ENDIF
SEEK NUMBER
IF FOUND ( )
ACTIVATE WINDOW  A
?? CHR(7)
 ? "هذا الرقم موجود فعلا"
 ? "من فضلك أعد الإدخال"
WAIT "إضغط أى مفتاح للإستمرار"
DEACTIVATE WINDOW  A
RETURN DEC
ENDIF
RETURN .NOT. DEC
```

٢ الوظيفة (Centering) ، تستخدم لوسطنة تعبير معين ...

```
FUNCTION CENTER
PARAMETERS STRING, WIDTH
RETURN ((WIDTH/2)-(LEN(STRING)/2))
```

وتستخدم هذه الوظيفة فى البرنامج كما يلى :

```
@ 5, CENTER (STRING, 80) SAY STRING
```

٣ الوظيفة (Altrim) ، تستخدم لإزالة كل الفراغات حول تعبير معين ...

```
FUNCTION ALTRIM
PARAMETERS STRING
RETURN (LTRIM(RTRIM(STRING)))
```

وتستخدم هذه الوظيفة فى البرنامج كما يلى :

```
@ 5,5 SAY ALTRIM (NAME)
```

٤ الوظيفة (AMPM) ، تستخدم لتحويل تعبير زمنى الى وضع ١٢ ساعة.

```
FUNCTION AMPM
PARAMETERS  T
RETURN IF(VAL(T)<12, T + "  am",;
IF(VAL(T) = 12, T + "PM",;
STR(VAL(T) - 12,2) + SULSTR (T,3) + "PM"))
```

وتستخدم فى البرنامج كما يلى :

```
String (12-hour found) = AMPM (<Time string >)
```

الفصل السابع

تنظيم الملفات

* تنظيم الملفات داخل كتالوج - فتح الكتالوج - إضافة ملف الى الكتالوج.
* إستخدام ملفات فى مشغل أقراص آخر.
* تنفيذ برنامج فى مشغل أقراص آخر.
* أمر استعراض الملفات.
* أمر استعراض محتويات الملفات.
* تعديل أسماء الملفات.
* نسخ الملفات.
* حذف الملفات.
* إغلاق الملفات.

الفصل السابع

تنظيم الملفات

١- تنظيم الملفات داخل كتالوج (Catalog) :

من المفضل أن تقوم بتوزيع ملفاتك على أكثر من كتالوج ، بحيث يحتوى كل كتالوج على مجموعة الملفات التى تشترك فى إنجاز مهام متجانسة ، ومن الممكن أن ينتمي ملف واحد لأكثر من كتالوج واحد .

٢- فتح الكتالوج :

لفتح كتالوج معين ، إكتب من نقطة الأوامر (Dot prompt) أو من خلال البرنامج ... SET CATALOG TO <filename > .

٣- إضافة ملف الى الكتالوج :

أ - إفتح الكتالوج الذى تريد إضافة الملف اليه كما سبق .

ب - إفتح الملف الى تريد إضافته الى الكتالوج ... Use <filename > .

٤- إستخدام ملفات فى مشغل أقراص (Drive) آخر :

لفتح الملف (Store dbf) الموجود على القرص المرن (Floppy disk) فى مشغل (A) - [Drive A] ، إكتب ... USE A: STORE .

٥- تنفيذ برنامج (.prg) فى مشغل أقراص آخر :

لنفترض أنك تريد تنفيذ البرنامج (PYRL .prg) الموجود على القرص المرن فى المشغل (Drive A) ، إكتب من نقطة الأوامر أو من خلال برنامج ...

```
.SET DEFAULT TO A:
.DO PYRL
```

٦- إستعراض الملفات (Dir)

لاستعراض أسماء الملفات وعدد السجلات وتاريخ آخر تعديل فى الملف والمساحة المستنفدة من القرص (Disk) لكل ملف بالبايت (Bytes) ، وعدد الملفات والمساحة الكلية المستنفدة من القرص لكل الملفات بالبايت .

أ - لاستعراض ملفات قاعدة البيانات (dbf.) فى الدليل الحالى ، إكتب من نقطة الأوامر ... Dir

ب - لاستعراض كل الملفات الموجودة فى الدليل الحالى ، إكتب من نقطة الأوامر ... *.* Dir

جـ - لاستعراض فئة معينة من الملفات :

مثلاً استعراض كل ملفات التشكيل (Format) الموجودة فى الدليل الحالى ، إكتب ... DIR * .FMT .

ويمكن استعراض كل الملفات التى تشترك فى حرف أو مجموعة حروف DIR AL ???.*

لإظهار كل الملفات التى تتكون أسماؤها من خمسة حروف وتبدأ بالحرفين (AL).

٧- الأمر LIST / DISPLAY

مشابه للأمر DIR فيما عدا أنه فى الأمر List / Display file يمكن إرسال الخرج (Output) للملف (File) أو للطابعة (Printer) .

مثال :

LIST FILES TO PRINTER

٨- تعديل أسماء الملفات

يمكنك تعديل إسم أى ملف الى أى إسم آخر بشرط أن يشتمل كل إسم ملف على الإمتداد (Extension) الخاص به .

مثال :

لإعادة تسمية الملف STORE .prg الى STOCK .prg ، إكتب ...

Rename store .PRG TO stock .PRG

٩- إستعراض محتويات ملف برنامج :

يمكنك استعراض محتويات برنامج أو إجراء باستخدام الأمر TYPE.

الصورة العامة:

Type<filename>[TO PRINTER/TO File<filename>]
[number]

أ - لاستعراض محتويات البرنامج على الشاشة يكتب ...

.Type <filename>

ب - لطباعة محتويات برنامج يكتب ...

Type <filename> TO PRINTER [number]

الاختيار [number] لكتابة رقم لكل سطر فى البرنامج.

جـ - لإرسال محتويات برنامج الى ملف يكتب ...

Type <filename> TO File <filename>

١٠- أمر النسخ Copy File

لنسخ الملف Store .dbf الى ملف آخر باسم Stock dbf ، إكتب ...

COPY FILE STORE .dbf TO STOCK .dbf

١١- النسخ للملفات لا تنتمى الى dBase IV

الشكل العام :

COPY TO <filename >[[TYPE]<file type >];
[Fields <field list >][<scope >][FOR <condition >];
[While <condition >]

يمكنك نسخ (Copy) حقول معينة بوضع أسماء هذه الحقول فى <field list > كذلك يمكنك نسخ سجلات معينة تقابل شرط محدد باستخدام [FOR <cond >] و [while <cond >] .

الإختيارات المتاحة فى (Type) هى :

١- [Delimited [with <delimiter >/Blank] .

وذلك للنسخ الى ملف مرمّز (مكوّد) باستخدام (ASCII) ويكون له الإمتداد (extension) الذى هو (.TXT)

أ – Delimited with <delimiters > ، للفصل بين حقول الملف بحرف واحد .
ب – Delimited with BLANK ، للفصل بين حقول الملف بمسافات خالية (Single space) .

٢- SDF (System data format)
للنسخ لملف مرمّز (مكوّد) باستخدام (ASCII) وله الإمتداد (.TXT) .

مثال :

```
.USE DEPART .DBF
.COPY TO DEPART .TXT SDF
```

فى هذا المثال يتم نسخ الملف (Depart .dbf) الى ملف مرمّز باستخدام (ASCII) وله الإمتداد (.TXT) .

٣- DBASE II
لنسخ ملف ينتمى الى dBase IV الى ملف ينتمى الى dBase II ، ويكون له الإمتداد (.db2) .

٤- DBMEMO3
لنسخ ملف قاعدة بيانات (.dbf) وملف حقل الذاكرة (.dbT) ينتمى الى dBase IV الى ملف ينتمى الىَ dBase III+ .

٥- RPD
لنسخ ملف ينتمى الى DBASE IV لملف آخر من النوع RPD .

٦- FW2 (Framework II .fw2)
لنسخ ملف DBase IV الى ملف آخر ينتمى الى Framework II ، وله الإمتداد (.FW2) .

٧- SYLK (Multiplan spreadsheet formula)
لنسخ ملف dBase IV الى ملف آخر ينتمى الى Multiplan spreadsheet formula ، وليس له امتداد .

٨- DIF (VisiCalc .dif)
لنسخ ملف dBase IV الى ملف آخر ينتمى الى VisiCalc ، وله الإمتداد (.dif) .

٩- WKS (Lotus 1-2-3 spreadsheet (.WKS))
لنسخ ملف dBase IV الى ملف ينتمى الى Lotus 1-2-3 ، الإصدار (1A) وله الإمتداد (.WKS) .

حذف الملفات Erase

يمكن حذف أى ملف موجود على الدليل (Directory) الحالى باستخدام الأمر:
Erase <filename >

أو الأمر :

Delete File <filename >

ويجب أن يحتوى إسم الملف (Filename) على الإمتداد (Extension) الخاص به .

إغلاق الملفات Close

أ - لإغلاق جميع الملفات يستخدم الأمر ...
CLOSE ALL

ب - لإغلاق ملفات قاعدة البيانات والفهارس (Index) وملفات التشكيل (Format files) يستخدم الأمر ...
. CLOSE DATABASES

جـ - لإغلاق ملف قاعدة البيانات المفتوح يستخدم الأمر ...
.USE

الفصل الثامن

إدخال وتعديل ومعالجة البيانات

* طرق تشكيل شاشة إدخال البيانات.
* إضافة بيانات من ملف لا ينتمى الى dBase IV.
* إستخدام النوافذ.
* الأمر SET DELIMETERS
* الوظيفة INKEY ( )
* الوظيفة READKEY ( )
* الوظيفة LASTKEY ( )
* تعديل البيانات - حذف السجلات.
* إستبدال محتويات حقل.
* ملامح متقدمة فى تعديل البيانات.
* معالجة البيانات - الأمر SUM - الأمر TO-TAL - الأمر CALCULATE.

الفصل الثامن

إدخال وتعديل ومعالجة البيانات

عند تعامل المستخدم مع ملفات قاعدة البيانات بإدخال البيانات أو بتعديلها ، يجب أن لا يكون التعامل مباشرا ، بل يجب أن يوجد وسيط (مصفوفة (Array) أو حقول ذاكرة متغيرة (Memvars)) بين المستخدم وقاعدة البيانات حيث يتم تخزين البيانات ، وعند التأكد من صحتها يتم نسخها الى ملفات قاعدة البيانات . كذلك يجب تهيئة شاشة إدخال البيانات وتشكيلها لتسهيل عمليات الإدخال والتعديل .

وسوف نستعرض الطرق المختلفة لتحقيق هذا الغرض ...

أولاً : إدخال البيانات

الأمر CREATE SCREEN

ينشئ شكلا للشاشة ملائم لإدخال البيانات عن طريق استخدام شاشة تصميم نماذج الأشكال (Forms design screen) ، وهو ينشئ ملفين لهما نفس الإسم ، أحدهما له الإمتداد (.scr) ، والآخر (.fmt) .

الصورة العامة :

CREATE / MODIFY SCREEN <filename >/?

الملف الذى له الإمتداد (.scr) يحتوى على معلومات عرض الشاشة .
والملف الذى له الإمتداد (.fmt) يحتوى على الأوامر لبناء شكل الشاشة .

عند التعديل فى سجلات (Records) ملف قاعدة بيانات يجب أولا أن تفتح الملف باستخدام الأمر ...

SET FORMAT TO <format >

وتستطيع غلق ملف التشكيل (Format file) بالأمر ...

SET FORMAT TO

العلامة (?) تعنى أنه إذا كان الكتالوج مفتوحا ، فإنه يستفسر عن كل ملفات التشكيل (.scr) التى لها صلة بملف قاعدة البيانات ؛ وإذا كان الكتالوج مغلقاً ، فإنها تعنى إظهار كل ملفات التشكيل (.scr) الموجودة على القرص المغنط (Diskette) لتختار منها الملف الذى تريد تعديله .

ملحوظة : يمكنك إنشاء ملف التشكيل (Format file) باستخدام الأمر ...

MODIFY COMMAND <filename .fmt >

على أن يعطى لإسم الملف الإمتداد (.fmt)
وفى هذا الملف يستخدم الأمرين :

@ ... SAY
@ ... GET

الأمر : @

يستخدم لإظهار البيانات أو إدخال البيانات فى شكل معين فى مواقع محددة على الشاشة .

الصورة العامة :

@ <row>, <col>[SAY<expression> GET <variable >[COLOR]

١- الأمر SAY ... @

يعرض محتويات حقل (Field) فى ملف قاعدة البيانات المفتوح أو حقل ذاكرة متغير (Memvar) سبق إنشاؤه ، أو جملة تعبيرية .

أمثلة :

1- USE FILE1
@ 5,5 SAY NAME

يعرض محتويات الحقل (Name) الموجود فى الملف (File1) .

2- @ 5,30 SAY MSAL

يعرض محتويات حقل الذاكرة المتغيرة (Msal) .

3- @ 7,5 SAY "IS THIS CORRECT ?"

يعرض الجملة السابقة فى الموقع المحدد (7,5) .

٢- الأمر CLEAR ... @

يستخدم لمسح جزء من الشاشة .

أمثلة :

1- @ 2,3 CLEAR

يمسح مستطيل من الشاشة بداية من الركن الأيسر العلوى هو (الصف الثانى ، والعمود الثالث) وحتى نهاية الشاشة .

2- @ 5,5 CLEAR TO 8,8

يمسح المستطيل الذى ركنه الأيسر العلوى هو (5,5) ، وركنه الأيمن السفلى هو (8,8) .

٣- الأمر @ ... TO

يستخدم لرسم خط أو لرسم مستطيل ...

أمثلة :

@ 3,5 TO 3,20 DOUBLE

يرسم خطاً مزدوجاً فى الصف الثالث من العمود الخامس ، وحتى العمود العشرين .

@ 2,2 TO 8,10

يرسم مستطيلا ركنه الأيسر العلوى (2,2) ، وركنه الأيمن السفلى (8,10) .

٤- الأمر @ ... FILL

يتيح تغيير لون مستطيل محدد على الشاشة .

مثال :

@ 4,5 FILL TO 8,10 COLOR R/W

يجعل الكتابة الموجودة داخل المستطيل المحددة إحداثياته باللون الأحمر على خلفية باللون الأبيض .

٥- الأمر @ ... GET ...READ

يستخدم لاستقبال البيانات من المستخدم .

وفيما يلى مثال يبين كيفية عمل برنامج تشكيل :

```
*  TEST .fmt

CLEAR
@ 1,20 SAY "LABOUR INFORMATION"
@ 2,22 TO 2,44
@4,5 SAY "ENTER LABOUR NAME"
@4,24 GET MNAME
@ 6,5 SAY "ENTER ADP:"
@ 6,16 GET MADD
@ 8,5 SAY "ENTER BD:"
@8,14 GET MBD
@10,5 SAY "QUALIFIED:"
@10,16 GET MQUAL
READ
```

مثال :

لإعداد نموذج شكل (Format) مناسب لإدخال بيانات عن الطلاب فى ملف (Student .dbf) ، شكل (٨ – ١) ، ينفذ البرنامج التالى :

```
CLEAR
@ 1,25 TO 6,61
@ 2,28 SAY "كشف بحالة الطالب الدراسية"
@ 3,0 SAY "  "
@ 3,31 SAY "عن نصف العام الأول"
@ 5,28 SAY "التاريخ  :  /  /  "
@ 7,3 SAY  " رقم الطالب: "
@ 7,14 GET Studentno PICTURE "999"
@ 9,3 SAY  "إسم الطالب : "
@ 9,14 GET studname PICTURE "XXXXXXXXXXXX"
@11,2 SAY  "الفصل : "
@ 11,13 GET class PICTURE "99"
@12,2 TO 17,70
@13,5 SAY  "اسم المادة"
@ 1318 SAY  "الطبيعة   الكيمياء   الرياضيات التقدير العام"
@ 14,2 TO 14,70
@15,5 SAY  "الدرجات"
@ 15,18 GET physics PICTURE "99.99"
@15,28 GET chemistry PICTURE "99.99"
@15,42 GET math PICTURE "99.99"
@ 15,64 SAY (physics+math+chemistry)/3 PICTURE "99.99"
@ 15,76 SAY "%"
@ 18,2 SAY  "مذكرات :
@ 18,10 TO 20,75
RETURN
```

كشف بحالة الطالب الدراسية
عن نصف العام الأول

التاريخ : / /

رقم الطالب : ١

إسم الطالب : هشام كمال عبد العال

الفصل : ١

إسم المادة	الطبيعة	الكيمياء	الرياضيات	التقدير العام
الدرجات	٩٠,٠٠	٨٠,٠٠	٩٠,٠٠	٨٦,٦٧٪

مذكرات:

شكل (٨ – ١)

من الممكن استخدام الأمر <var> GET ... @ مع عدة اختيارات :

١- @ ... GET <var >[window <window name >]

تستخدم عند التعامل مع حقول الذاكرة (Memo fields) لفتح نافذة مستقلة (سبق تعريفها) لتعديل محتويات حقل الذاكرة ، وذلك بوضع المؤشر على حقل الذاكرة (memo field) واضغط (ctrl-Home) لفتح النافذة ثم اضغط (ctrl-End) لغلقها .

٢- @ ... GET <var >[PICTURE <expc >

تحدد شكل البيانات باستخدام دوال تشكيل (Format fuction) أو رموز قوالب (Tamplete symbol) .

__مثال :__ @ 5,5 SAY SALARY PICTURE "999.99"

يُظهر (Salary) بشكل رقمى فى حدود ٣ أرقام صحيحة ورقمين عشريين .

__ملحوظة :__ الرمز (9) ليس قيمة وإنما رمزاً لعرض الأرقام فقط ، والعدد الذى يكرر به يرمز لعدد مواضع الأرقام الممكن قبولها .

__مثال :__ @ 5,5 GET Mtot PICTURE "@$999.99"

يُظهر القيمة مع علامة الدولار ($) .

__ملحوظة :__ $ @ ، هى دالة تشكيل .

دوال التشكيل (FORMAT FUNCTION)

من الممكن استخدام معظم دوال التشكيل مع PICTURE أو FUNCTION أو الوظيفة TRANSFORM .

أ - إذا استُخدمت دوال التشكيل مع Picture ، يجب أن يوضع الرمز (@) فى بداية العبارة (CLAUSE) .

ب - إذا استُخدمت دوال التشكيل مع FUNCTION فلن تحتاج لوضع الرمز (@) .

جـ - إذا استخدمت كل من دوال التشكيل (Format Function) مع رموز القوالب (Tampletes) فيجب أن تضع مسافة خالية (Space) بينهما .

بعض دوال التشكيل الهامة FORMAT FUNCITON

! يحوّل الحروف الى وضع الحروف الكبيرة .
^ يُظهر الأرقام فى الشكل العلمى (Scientific format) .
A يقبل ويُظهر الحروف فقط .
B يضبط وضع نص فى حقل ما من ناحية اليسار .
E يُظهر التاريخ فى الشكل الأوربى .
I يضبط وضع نص فى حقل ما فى الوسط .
J يضبط وضع نص فى حقل ما فى اليمين .
L يعرض الأصفار يسار العدد .
T يزيل المسافات الخالية من الحقل (Field) .

مثال :

```
Mname = space (20)
@ 5,5 SAY "ENTER THE NAME:" GET MNAME FUNCTION "!"
READ
```

فى هذا المثال يتم تحويل الحروف الصغيرة الى الحروف الكبيرة باستخدام دالة التشكيل (!) .

بعض رموز قوالب التشكيل الهامة

TAMPLETE SYMBOLS

!	يحول الحروف الى الحروف الكبيرة .
#	يسمح فقط بالأرقام والمسافات الخالية والعلامات .
*	يظهر علامة (*) مكان الأصفار يسار العدد .
,	يبين ماإذا كانت توجد أرقام يسار الفاصلة (Comma)
.	يحدد موقع العلامة العشرية .
9	يسمح فقط بالأرقام .
A	يسمح فقط بالحروف .
L	يسمح فقط بالبيانات المنطقية .
N	يسمح بالحروف والأرقام .
X	يسمح بالحروف والأرقام والعلامات والمسافات الخالية .
Y	يسمح فقط بالحروف المنطقية (N, n, Y, y) .

مثال :

```
MNO = 0
@ 5,5 SAY "ENTER THE ITEM NUMBER:";
GET MNO PICTURE "9999"
```

فى هذا المثال يتم قبول أرقام فقط ، ولا تزيد عن أربعة أرقام .

مثال آخر :

```
@ 7,3 SAY "رقم الطالب : "
@ 7,14 GET studentno PICTURE "999"
@ 9,3 SAY "اسم الطالب : "
@ 9,14 GET studname PICTURE "XXXXXXXXXXXXX"
@ 11,2 SAY "الفصل : "
@ 11,13 GET class PICTURE "99"
```

٣- @ ... GET <var >[Range [<low>][,<high>]]

يحدد الحدين الأدنى والأقصى الممكنين للأرقام والتاريخ . ويمكن كذلك تحديد الحد الأدنى فقط أو الحد الأقصى فقط .

مثال : @ 3,5 GET MNO Range 30,90

يقبل فقط الأرقام من ٣٠ الى ٦٠

مثال : @ 5,5 GET MDATE Range, {16/11/89}

يقبل فقط التاريخ الذى يقع قبل ١٦ نوفمبر ٨٩

٤- @ ... GET <var >[VALID <condition>[Error <expc>]]

يستخدم للتأكد من مطابقة البيانات المدخلة للشرط الذى تحدده ، فإن لم تكن مطابق أظهرت الرسالة التالية :

Editing condition not satisfied

أو أى رسالة إخرى تحددها .

مثال :

@ 5,5 SAY "Is this right ?" GET ANSWER VALID ANSWER $ "Y/N" ERROR; "PRESS Y or N only"

فى هذا المثال يتم قبول حرفى Y أو N فقط ، فإذا أُدخل أى حرف آخر غيرهما ظهرت الرسالة : Press Y or N only .

٥- @ ... GET <var>[when <cond>][ERROR <expc>]]

تحدد متى يمكن للمستخدم أن يقوم بتعديل حقل ما (Field) ، فإذا تحقق الشرط تحرك المؤشر للحقل المراد تعديله ، وإذا لم يتحقق الشرط تعدى المؤشر هذا الحقل للحقل الذى يليه .

مثال :

@ 5,5 GET NAME WHEN No#0 ERROR "enter the number before the name"

يمنع إدخال حقل الإسم طالما كان حقل (No) = صفر .

مثال :

البرنامج التالى يقوم بإعداد نموذج شكل لإدخال بيانات الصنف فى برنامج مراقبة المخزون :

```
PROCEDURE ASCREEN
1  @ 1,1 TO 23,78 DOUBLE
2  @2,30 TO 4,50
3  @3,32 SAY "بيانات الصنف"
4  @ 6,4 SAY "رقم الصنف :" ;
5  GET MNO PICT "9999"VALID DUPLICATE (MNO);
6  MESS "إدخل رقم الصنف ..."
7  @ 8,4 SAY "اسم الصنف :" GET MNAME WHEN MNO >0;
8  MESS "إدخل اسم الصنف ..."
9  @ 10,4 SAY "الوحدة: " ;
10 GET MUNIT RANG 1,5 WHEN MNO >0;
11 MESS "كم=1.. كم=2.. طن=3.. بالعدد=4.. كج=5"
12 ERRO "من فضلك إدخل من 1 الى 5 فقط"
13 @ 12,4 SAY "النوع :";
14 GET MTYP RANG 1,17 WHEN MNO >0;
15 MESS "إدخل رقم من 1 الى 17 ..."
16 @14,4 SAY "مستوى الطلب :" GET MCL WHEN MNO >0;
17 MESS "إدخل حد الطلب ..."
18 @16,4 SAY "المستوى الحرج :" GET MDL WHEN MNO >0;
19 MESS "إدخل المستوى الحرج ..."
20 RETURN
```

١- السطر الأول لعمل إطار (Frame) من الصف (١) والعمود (١) الى الصف (٢٣) والعمود (٧٨) .

٢- السطر الثانى لعمل برواز للعنوان الرئيسى .

٣- السطر الرابع لاستقبال رقم الصنف من المستخدم الى حقل ذاكرة متغير (MNO) ؛ Pict "9999" --> تعنى أن هذا الحقل يقبل أرقاما فقط لا تزيد عن أربعة أرقام ؛

Valid Duplicate (MNO) <---- (Valid) تعنى تطبيق شرط معين على ما يدخله المستخدم ، ولا يتم قبوله مادام لا يحقق هذا الشرط ، وهنا يتم تطبيق إحدى الوظائف التى يُعرِّفها المستخدم (DUF) وهى الوظيفة (Duplicate) التى تتحقق من عدم تكرار نفس رقم الصنف خلال الملف.

٤- لاحظ حقل إدخال إسم الصنف (سطر ٧) ؛

when MNO >0 <—— تعنى أنه إذا لم يقم المستخدم بادخال رقم الصنف أولا ، فإن المؤشر سوف يتعدى حقل إدخال الإسم ، ولا يتاح للمستخدم إدخال اسم الصنف .

" ... إدخل اسم الصنف" MESS <--- تعنى أنه سيتم عرض رسالة : "إدخل إسم الصنف ... " أسفل الشاشة عند وقوف المؤشر عند حقل اسم الصنف .

٥- لاحظ فى حقل الوحدة (سطرا ٩ و ١٠) ؛

(Rang 1,5) تعنى أنه سيتم قبول رقم تتراوح قيمته من ١ الى ٥ فقط.

وعند تنفيذ البرنامج السابق تعرض شاشة إدخال البيانات التالية.

بيانات الصنف

رقم الصنف :
إسم الصنف :
الوحدة :
النوع :
مستوى الطلب :
المستوى الحرج :

إدخل رقم الصنف ..

شكل (٨ - ٢)

إضافة بيانات من ملف لا ينتمى الى dBase IV

باستخدام الأمر APPEND FROM ، يمكنك إضافة بيانات من ملف لا ينتمى الى dBase IV الى نهاية ملف قاعدة بيانات مفتوح .

ملحوظة : الأمر Append From يمكّنك أيضا من إضافة بيانات من ملف قاعدة بيانات ينتمى الى dBase IV الى نهاية ملف قاعدة بيانات مفتوح .

الشكل العام :

```
APPEND FROM <filename >/ [[TYPE]<file type>];
[.FOR <condition >]
```

الإختيارات المتاحة فى [TYPE] هى :

١- Delimited [with <delimiter>/Blank]

للإضافة من ملف مرمّز باستخدام (ASCII) وله الإمتداد (.TXT) .

أ - delimited with <delimiter> ، يستخدم حرفا محددا فى الفصل بين حقول الملف .

ب - delimited with Blank ، تستخدم مسافة خالية (single space) فى الفصل بين حقول الملف .

٢- SDF (System Data Format)

للإضافة من ملف مرمّز باستخدام (ASCII) وله الإمتداد (.TXT) .

مثال :

```
USE DEPART
APPEND FROM DEPART .TXT TYPE  SDF
```

٣- DBASE II

للإضافة من ملف ينتمى الى dBase II .

٤- RPD

للإضافة من ملف ينتمى الى RapidFile™

٥- DIF

للإضافة من ملف ينتمى الى VisiCalc ، وله الإمتداد (.dif) .

٦- FW2
للإضافة من ملف ينتمى الى Framework II™ وله الإمتداد (.FW2) .
٧- SYLK (Multiplan Spreadsheet Formula)
للإضافة من ملف ينتمى الى (... Multiplan) وليس له امتداد .

ملحوظة : الأمر Append from يكافئ الأمر IMPORT .

والشكل العام للأمر Import :

IMPORT FROM <filename>[TYPE]PFS/DBASE II/FW2/RPD/WK1

والإختيارات المتاحة فى [TYPE] هى :

أ - PFS
ب - DBASE II
جـ - FW2
د - RPD
هـ - WK1

إستخدام النوافذ WINDOWS

النافذة هى عبارة عن مستطيل يأخذ حيزا من الشاشة تُعرض فيه البيانات أو الرسائل ، وتستطيع استخدام أكثر من نافذة واحدة فى نفس الوقت .

إنشاء النوافذ

١- أولا يجب تعريفها وتحديد الإحداثيات (Co-ordinations) كالتالى :
DEFINE WINDOW <name> FROM <coords> TO <coords>
٢- عند الإستخدام ، إكتب الأمر... ACTIVATE WINDOW <name>
٣- عند الرغبة فى إزالة الإطار من الشاشة ، إكتب ...
DEACTIVATE WINDOW <name> .

مثال :

```
Des = " "
DEFINE WINDOW M FROM 10,10 TO 15,70
ACTIVATE WINDOW M
@ 2,1"is this correct ? (Y/N)" GET DES
READ
DEACTIVATE WINDOW M
```

إنشاء نافذة لتعديل حقول الذاكرة (Memo Fields)

١- عرّف النافذة ...

```
DEFINE WINDOW <name > FROM <coords> TO <coords>
```

٢- إكتب :

```
SET WINDOW OF MEMO TO <window name>
```

تحريك النافذة على الشاشة

بإمكانك نقل نافذة من موقع حالى الى موقع آخر على الشاشة باستخدام الأمر:

```
MOVE WINDOW <window name> TO <row>, <column>/BY
<delta row>, <delta column>
```

مثال -١

لتحريك نافذة M بمقدار ٣ أعمدة لليمين و ٤ أعمدة لأسفل ...

```
MOVE WINDOW M BY 4,3
```

مثال - ٢

لتحريك نافذة M لموقع جديد على الشاشة ...

```
MOVE WINDOW M TO 10,10
```

الإحتفاظ بالنافذة (Window) بملف بالقرص الممغنط (Disk)

بإمكانك الاحتفاظ بنافذة أو أكثر داخل ملف ليمكنك استعادتها مرة أخرى للإستخدام ، كالتالى :

SAVE WINDOW <window name list>/ALL TO <filename>

Window name list ، للإحتفاظ بمجموعة نوافذ تحدد أسماؤها .

ALL ، للإحتفاظ بجميع النوافذ الموجودة فى ذاكرة الحاسب داخل الملف.

إسترجاع النوافذ

لاسترجاع النوافذ التى سبق وأن حفظتها داخل ملف على القرص الممغنط الى ذاكرة الحاسب (RAM) ...

RESTORE WINDOW <window name list>/ALL FROM <filename>

أ – <window name list> ، لاسترجاع نوافذ معينة .

ب – ALL ، لاستعادة جميع النوافذ .

الخروج من النوافذ دون إزالتها من الشاشة

أحياناً قد ترغب فى الخروج من النافذة واستخدام بقية الشاشة دون إزالتها من عليها ؛ يمكنك ذلك بكتابة الأمر ACTIVATE SCREEN.

الأمر SET INTENSITY ON/OFF

يتحكم فى طريقة إظهار البيانات الداخلة بواسطة الأمر GET ، فعندما يكون على الوضع ON تكون خلفية الشاشة مضيئة ، وعندما يكون على OFF تكون خلفية الشاشة غير مضيئة بينما الكتابة مضيئة .
الضبط المسبق (Default) هو ON .

الأمر SET DELIMETERS

١- الصورة العامة ... SET DELIMETERS ON/OFF
تستخدم لتحديد ما إذا كانت علامات تحديد بداية ونهاية الحقول ستستخدم أم لا .

٢- SET DELIMETERS TO <character>
تحدد شكل العلامات التى ستستخدم لتحديد بداية ونهاية الحقول.

مثال :

لوضع العلامة ([) قبل بداية الحقل ، والعلامة (]) بعد نهاية الحقل يكتب الآتى :

```
SET DELIMETERS TO "[]"
SET DELIMETERS ON
```

الوظيفة INKEY ( )

هذه الوظيفة تعطى رقما صحيحا يقابل آخر مفتاح تم الضغط عليه ، وهذا الرقم يقع بين صفر ، ٢٥٥ ، وهى الأرقام المقابلة للمفاتيح حسب الرمز الأمريكى القياسى (ASCII) ، وتستخدم هذه الوظيفة عادة فى البرامج كشرط للتفريع (Branching) وتتيح استعمال مفاتيح الوظائف (Functions keys) ومفاتيح حركة المؤشر خلال البرنامج ، وفى حالة عدم ضغط أى مفتاح ، فإن INKEY ( ) تعطى القيمة صفر. وعادة تستعمل مع القوائم .

مثال :

```
1 CLEAR
2 SET TALK OFF
3 DO WHILE .T.
4 ? "                MAIN MENU"
5 ? "                ========="
6 ?
7 ?
8 ? "        A = APPEND "
9 ? "        B = EDIT"
10 ? "       C = DELETE"
11 ? "       X = EXIT"
12 ?
13 ? "PLEASE ENTER YOUR CHOICE:"
14 DO WHILE I = 0
15 I = INKEY ( )
16 DO CASE
17 CASE CHR(I)$"Aa"
18 DO PROG1
19 CASE CHR (I)$"Bb"
20 DO PROG2
21 CASE CHR (I)$"Cc"
22 DO PROG3
23 CASE CHR (I)$"Xx"
24 CLOSE DATABASES
25 QUIT
26 ENDCASE
27 ENDDO
28 ENDDO
29 RTURN
```

١- من السطر (١٤) الى (٢٧) حلقة دوارة مادام المستخدم لم يقم بضغط أى مفتاح فأن قيمة (I) تظل = صفرا.

٢- عند قيام المستخدم بضغط المفتاح A فإنه يتم التفريع لتنفيذ البرنامج الفرعى PROG1 .

وعند تنفيذ هذا البرنامج فإنه يتم عرض القائمة التالية:

MAIN MENU
==========

A = APPEND
B = EDIT
C = DELETE
X = EXIT

PLEASE ENTER YOUR CHOICE:

شكل (٨ – ٣)

جدول ( ) INKEY

المفتاح	الرقم	المفتاح	الرقم
ctrl - B (ctrl - →)	٢	ctrl - A (HOME)	١
ctrl - D (→)	٤	ctrl - C (Pg Dn)	٣
ctrl - F (ctrl - END)	٦	ctrl - E (↑)	٥
ctrl - H	٨	ctrl - G (Del)	٧
ctrl - J	١٠	ctrl - I (TA B)	٩
ctrl - L	١٢	ctrl - K	١١
ctrl - N	١٤	ctrl - M	١٣
ctrl - P	١٦	ctrl - O	١٥
ctrl - R (Pg Up)	١٨	ctrl - Q	١٧
ctrl - T	٢٠	ctrl - S (←)	١٩
ctrl - V (INS)	٢٢	ctrl - U	٢١
ctrl - X (↑)	٢٤	ctrl - W (ctrl - END)	٢٣
ctrl - Z	٢٦	ctrl - Y	٢٥
F1 (ctrl - \ )	٢٨	ESC (ctrl - [ )	٢٧
ctrl - ^ (ctrl - Pg Dn)	٣٠	ctrl -] (ctrl - HOME)	٢٩
spacebar	٣٢	ctrl - (ctrl - Pg Up)	٣١
Backtab	-٤٠٠	ctrl - Back space	-٤٠١
ctrl - ↵	-٤٠٢	F2	-١
F3	-٢	F4	-٣
F5	-٤	F6	-٥
F7	-٦	F8	-٧
F9	-٨	F10	-٩

الوظيفة READKEY ( )

تعطى رقما صحيحا يقابل المفتاح الذى تم الضغط عليه للخروج من شاشة تنفيذ أوامر الشاشة الكاملة [READ, APPEND, BROWSE, CHANGE, EDIT] ، وهذا الرقم يتغير إذا ما تغيرت أى بيانات خلال العمل مع أوامر الشاشة الكاملة.

كل ضغطة مفتاح تعطى واحد من احتمالين :

أ - إذا تغيرت البيانات فإن الشفرة تكون بين ٥ ، ٣٦

ب - إذا لم تتغير البيانات فإن الشفرة تزيد عن ٢٥٦

مثال :

لمعرفة ما إذا كان حقل الذاكرة المتغير (Msal) قد تغيرت محتوياته ولتخزينه فى حقل الملف إذا أحدث فيه تغيير ...

```
IF READKEY ( ) >= 256
REPLACE SAL WITH MSAL
ENDIF
```

جدول ( ) READKEY

المفتـــاح	الرقم الكودى (بعد التحديث)	الرقم الكودى (بدون تحديث)
ctrl - S ←, ctrl - H	٢٥٦	صفر
Backspace	٢٥٦	–
ctrl - D , →, ctrl - L	٢٥٧	١
ctrl - E , ctrl - K , ↑	٢٦٠	٤
ctrl - J , ctrl - X , ↓	٢٦١	٥
ctrl - R , Pg Up	٢٦٢	٦
ctrl - Q , \ESC	–	١٢
ctrl - W , ctrl - END	٢٧٠	–
↵ , ctrl - M	٢٧١	١٥
↵ , ctrl - M	–	١٦
عند بداية سجل فى حالة (Append)		
ctrl - HOME	٢٨٩	٣٣
ctrl - Pg Up	٢٩٠	٣٤
ctrl - Pg Dn	٢٩١	٣٥
F1	٢٩٢	٣٦

مثال :

فى حالة عدم حدوث أى تعديل خلال شاشة التعديل وهى الشاشة التى تستخدم الأمر

@... SAY ... GET
READ

فإن قيمة READKEY ( ) تكون أقل من ٢٥٦ ، وفى هذه الحالة يتم الخروج من الحلقة الدوارة (Do while .T.) باستخدام الأمر EXIT .

```
DO WHILE .T.
STORE 0 TO MNO,MCLMDL
MUNIT = 0
MNAME = SPACE (40)
MTYP = 0
@ 1,1 TO 23,78 DOUBLE
@ 2,30 TO 4,50
@ 3,32 SAY "EDIT ITEM"
@ 4,4 SAY "ITEM NUMB:" GET MNO PICT "99999999"MESS "ENTER THE ITEM NUMBER..."
@ 6,4 SAY "ITEM NMAE:" GET MNAME MESS "ENTER THE ITEM NAME..."
@ 8,4 SAY "UNIT     :" GET MUNIT RANG 1,5 MESS "ENTER THE UNIT FROM 1 -> 5 ONLY ..."
@ 10,4 SAY "TYPE     :" GET MTYP MESS "ENTER THE TYPE FROM 1 -> 17 ONLY..."
@ 12,4 SAY "DEMAND LVL:" GET MCL MESS "ENTER DEMAND LVL..."
@ 14,4 SAY "CRITICAL LVL:" GET MDL MESS "ENTER CRIT, LVL..."
READ
IF READKEY ( ) < 256
CLEA
EXIT
RETURN TO MASTER
ENDI
ENDDO
```

الوظيفة LASTKEY ( )

تعطى القيمة المقابلة لآخر مفتاح تم الضغط عليه حسب الرمز الأمريكى AS-CII وتعطى نفس القيمة مثل الوظيفة READYKEY ( ) ، وتستخدم فى البرنامج للحصول على القيمة المقابلة للمفتاح الذى يضغط عليه المستخدم ، وبناء على قيمة هذا المفتاح يتم إنجاز أمر أو إجراء ما يقابل هذه القيمة.

مثال:

فى البرنامج التالى عند قيام المستخدم بضغط مفتاح (F3) فإنه يعرض عليه من خلال سطر الرسالة أسفل الشاشة العمل الذى فعله بالأمس ، وعند ضغط (F4) فإنه يعرض عليه من خلال سطر الرسالة ما يجب أن يفعله بالغد.

```
CLEAR
SET TALK OFF
SET MESSAGE TO " "
DEC = 0
@ 10,10 SAY "PRESS F3 TO (YESTERDAY) F4 TO (TOMORROW)" GET DEC PICTURE "99"
READ
IF LASTKEY ( ) = -2
? CHR (7)
SET MESSAGE TO "YOU WENT TO BANK MISR"
X = INKEY (5)
ENDIF
IF LASTKEY ( ) = -3
? CHR (7)
SET MESSAGE TO "YOU MUST TRAVEL TO PARIS"
X = INKEY (5)
ENDIF
RETURN
```

ثانيا : تعديل البيانات

١- حذف السجلات (Delete records)

يتم ذلك بإجرائين هما :

أ - باختيار السجلات التى تريد حذفها باستخدام الأمر DELETE .

ب - بحذفها نهائيا من الملف باستخدام الأمر PACK .

الأمر DELETE

الشكل العام:

DELETE [<scape>][FOR<comd>][while <comd>]

مثال :

لحذف السجلين الأولين فى الملف (Aly .dbf)

```
.DISPLAY ALL
Record #     NUM NAME     ADD          TEL
   1          1 ESLAM     NASR CITY    2618438
   2          2 ADEL      NEW CAIRO     766400
   3          3 ASHRAF    ZAHRAA       6543890
   4          5 OSAMA     MAADI         765990
   5          6 AHMED     HELWAN        765499
. USE ALY ORDER NUM
. DELETE NEXT 2
. DISPLAY ALL
Record #     NUM NAME     ADD          TEL
   1*         1 ESLAM     NASR CITY    2618438
   2*         2 ADEL      NEW CAIRO     766400
   3          3 ASHRAF    ZAHRAA       6543890
   4          5 OSAMA     MAADI         765990
   5          6 AHMED     HELWAN        765499
```

لاحظ وجود العلامة (*) على السجلين الأول والثانى .

ولحذف سجل فى الملف إذا علمت رقم السجل ...

```
. GOTO 4
. DISPLAY
Record #        NUM NAME      ADD             TEL
     4          5  OSAMA      MAADI        765990
. DELETE
. DISPLAY ALL
Record #        NUM NAME      ADD             TEL
     1          1  ESLAM      NASR CITY   2618438
     2          2  ADEL       NEW CAIRO    766400
     3          3  ASHRAF     ZAHRAA      6543890
     4*         5  OSAMA      MAADI        765990
     5          6  AHMED      HELWAN       765499
```

لاحظ وجود العلامة (*) على السجل الرابع .

لحذف سجلات فى الملف تقابل شرطا محددا ...

```
.DISPLAY ALL
Record #        NUM NAME      ADD             TEL
     1          1  ESLAM      NASR CITY   2618438
     2          2  ADEL       NEW CAIRO    766400
     3          3  ASHRAF     ZAHRAA      6543890
     4          5  OSAMA      MAADI        765990
     5          6  AHMED      HELWAN       765499
. DELETE FOR NUM > 3
. DISPLAY ALL
Record #        NUM NAME      ADD             TEL
     1          1  ESLAM      NASR CITY   2618438
     2          2  ADEL       NEW CAIRO    766400
     3          3  ASHRAF     ZAHRAA      6543890
     4*         5  OSAMA      MAADI        765990
     5*         6  AHMED      HELWAN       765499
```

لاحظ وجود العلامة (*) على السجلين (٤) و (٥) اللذين يحققان شرط أن(NUM) أكبر من (٣) .

وعند استعراض سجلات الملف باستخدام LIST أو DISPLAY ALL ستجد أن علامة (*) وضعت فى بداية السجل الذى اختير للحذف . ولإزالة هذه العلامة من جميع سجلات الملف ، إستخدم الأمر RECALL ALL .

مثال :

```
.DISPLAY ALL
Record #     NUM NAME     ADD          TEL
   1*        1 ESLAM      NASR CITY    2618438
   2*        2 ADEL       NEW CAIRO     766400
   3         3 ASHRAF     ZAHRAA       6543890
   4         5 OSAMA      MAADI         765990
   5         6 AHMED      HELWAN        765499
 RECALL ALL
. DISPLAY ALL
Record #     NUM NAME     ADD          TEL
   1         1 ESLAM      NASR CITY    2618438
   2         2 ADEL       NEW CAIRO     766400
   3         3 ASHRAF     ZAHRAA       6543890
   4         5 OSAMA      MAADI         765990
   5         6 AHMED      HELWAN        765499
```

ولإزالة العلامة (*) من سجل محدد ، حرّك مؤشر السجلات (Pointer) لهذا السجل باستخدام الأمر GOTO ، ثم اكتب RECALL .

مثال :

```
. USE STORE
. GOTO 20
. RECALL
```

حذف السجلات (المختارة للحذف) نهائيا من الملف

يتم ذلك باستخدام الأمر PACK ليتم حذف جميع السجلات التى سبق اختيارها للحذف باستخدام الأمر DELETE .

مثال :

```
.DISPLAY ALL
Record #   NUM NAME     ADD          TEL
    1      1  ESLAM     NASR CITY    2618438
    2      2  ADEL      NEW CAIRO     766400
    3      3  ASHRAF    ZAHRAA       6543890
    4      5  OSAMA     MAADI         765990
    5*     6  AHMED     HELWAN        765499
. PACK
. DISPLAY ALL
Record #   NUM NAME     ADD          TEL
    1      1  ESLAM     NASR CITY     897655
    2      2  ADEL      NEW CAIRO     766400
    3      3  ASHRAF    ZAHRAA       6543890
    4      5  OSAMA     MAADI         765990
```

لاحظ اختفاء السجل (٥) بعد استخدام الأمر PACK .

حذف جميع سجلات الملف

لحذف جميع سجلات ملف مفتوح ، إكتب ZAP ؛ وهذا الأمر يكافئ كلا الأمرين DELETE ALL, PACK .

مثال :

```
. DISPLAY ALL
Record #      NUM NAME     ADD            TEL
     1        1  ESLAM     NASR CITY      2618438
     2        2  ADEL      NEW CAIRO      766400
     3        3  ASHRAF    ZAHRAA         6543890
     4        5  OSAMA     MAADI          765990
. ZAP
zap C:/DBASE/ALY .DBF? (Y/N)  yes
. DISPLAY ALL
```

لاحظ اختفاء جميع سجلات الملف .

تعديل محتويات حقل :

من المعتاد أثناء تعديل بيانات الحقول فى الملف إستقبال هذه التعديلات التى يريدها المستخدم فى حقول ذاكرة متغيرة (Memory Variables) ثم بعد التأكد من صحة تلك التعديلات يتم استبدال محتويات الحقول فى ملف قاعدة البيانات بهذه البيانات ، ويتم ذلك الإستبدال باستخدام الأمر REPLACE .

مثال :

```
. USE ALY
. DISPLAY
Record #      NUM NAME     ADD            TEL
     1        99 AMAL      AGOUZA         897789
. REPLACE NUM WITH NUM + 1
. DISPLAY
Record #      NUM NAME     ADD            TEL
     1        100 AMAL     AGOUZA         897789
```

لاحظ أن قيمة الحقل (NUM) قد تم استبداله بالقيمة ١٠٠.

ملامح متقدمة فى تعديل البيانات

[١] أحيانا قد ينقطع التيار الكهربائى أثناء القيام بالتعديل الدورى فى بيانات ملفات قاعدة البيانات ، مما يعرّض هذه البيانات التى ربما بُذل فيها الكثير من الوقت والجهد ، للفقد.

وقد قدمت dBase IV إمكانية جديدة وهى إحدى دوال (Set Options) ...

SET AUTOSAVE on/OFF

ففى حالة ON يتم حفظ بيانات الملف المعدلة تلقائيا بالقرص الممغنط بمجرد تغير الوسيط الخازن (Buffer) . الوضع المسبق لهذه الدالة هو الوضع OFF وذلك لإعطاء إمكانية إلغاء التعديلات التى يحدثها المستخدم فى الملف .

ولكن الأمر مختلف عادة فى البرنامج ذلك لأن تعديل البيانات فى الملف لا يتم بصورة مباشرة ، ولكن يتم أولا استقبال التعديلات التى يريدها المستخدم وتخزينها داخل مصفوفة ذاكرة (Array) أو حقول ذاكرة متغيرة (Memvar) وعند التأكد من صحتها يتم استبدال محتويات حقول الملف بمحتويات هذه الحقول المتغيرة ، ولذلك فمن الأفضل أن توضع هذه الدالة فى البرنامج فى الوضع ON .

[٢] أثناء عملية تعديل أو إضافة بيانات فى ملف قاعدة البيانات خلال برنامج التعديل ، قد يحدث خطأ ما يؤدى الى أن يتم التعديل فى البيانات بشكل جزئى . فى هذه الحالة قد ترغب فى إلغاء التعديل واستعادة البيانات الأصلية قبل عملية التعديل . وقد أعطت dBase IV إمكانية جديدة لاستعادة البيانات الأصلية مرة أخرى ، وإلغاء التعديل أثناء عملية تعديل البيانات وذلك باستخدام الأمر ...

```
BEGIN TRANSACTION [<path name>]
<transaction commands>
END TRANSACTION
```

والأمر ROLLBACK

مثال :

```
CLEAR
USE PAYROLL
ON ERROR ROLLBACK
BEGIN TRANSACTION
REPLACE TAX WITH TAX + TAX * 0.1
END TRANSACTION
```

فى هذا المثال تم استخدام الأمر (ON ERROR) لجعل الأمر (ROLLBACK) ينفذ تلقائيا بمجرد حدوث خطأ أثناء عملية التعديل. والأمر (ROLLBACK) يعنى استعادة جميع بيانات ملف قاعدة البيانات والفهارس لحالتها الأصلية قبل عملية التعديل.

مثال آخر :

```
CLEAR
USE PAYROLL
BEGIN TRANSACTION
REPLACE TAX WITH TAX + TAX * 0.1
END TRANSACTION
IF .NOT. COMPLETE ( )
ROLLBACK
ENDIF
```

فى هذا المثال يتم اختبار ما إذا كانت عملية التعديل (Transaction) قد انتهت وتم تنفيذ الأمر (End transaction) أم لا ، وذلك باستخدام الوظيفة Completed ( ) ؛ فإذا لم تكن عملية التعديل قد انتهت ، فيتم تنفيذ الأمر (Rollback) الذى يستعيد ملف قاعدة البيانات والفهارس كما كانت فى الأصل قبل عملية التعديل .

توجد بعض الأوامر التى لا يُسمح بها مع استخدام Begin/End transaction ، وهى :

1- CLEAR ALL
2- CLOSE ALL/DATABASE/INDEX
3- DELETE FILE
4- ERASE
5- INSERT
6- MODIFY STRUCTURE
7- PACK
8- ZAP

ثالثاً : معالجة البيانات

جمع البيانات الرقمية (SUMMING)

لإيجاد حاصل جمع الحقول الرقمية وتخزينها داخل حقل ذاكرة متغير (Memvar) أو داخل عناصر مصفوفة ذاكرة (Array) ...

* باستخدام الأمر SUM

الشكل العام :

```
SUM [<expN list >][TO <memvar list >/TO ARRAY<array name>]
[<scape>][FOR <cond>][WHILE <cond>]
```

مثال :

لإيجاد حاصل جمع Price وتخزينه داخل حقل ذاكرة متغير MPrice

```
. USE Z
. DISPLAY ALL
Record #          NO ITEM            PRICE        QUANT
      1           1 LAMP              2.00            8
      2           2 RADIO            60.00           80
      3           3 T.V.            900.00           70
      4           4 RECORDER        300.00           50
. SUM PRICE TO MPRICE
. ? MPRICE
       1262
```

مثال :

لإيجاد حاصل جمع Price , Quant وتخزينه داخل عناصر مصفوفة (M)

```
. USE Z
. DISPLAY ALL
Record #          NO ITEM            PRICE        QUANT
      1           1 LAMP              2.00            8
      2           2 RADIO            60.00           80
      3           3 T.V.            900.00           70
      4           4 RECORDER        300.00           50
. SUM PRICE, QUANT TO ARRAY M
      4 records summed
    PRICE    QUANT
     1262      208
```

مثال :

لجمع جميع الحقول الرقمية فى الملف وعدم تخزينها داخل حقول ذاكرة متغيرة أو مصفوفة ، ثم عرض حاصل الجمع على الشاشة ...

```
. USE Z
. DISPLAY ALL
Record #      NO ITEM          PRICE     QUANT
      1        1 LAMP           2.00         8
      2        2 RADIO         60.00        80
      3        3 T.V.         900.00        70
      4        4 RECORDER     300.00        50
. SUM
    4 records summed
  NO  PRICE   QUANT
  10   1262     208
```

* الجمع باستخدام الأمر TOTAL

يجمع كل الحقول الرقمية (أو بعضا منها) فى الملف المفتوح ثم يضع حاصل جمع هذه الحقول داخل ملف جديد ، ويجب أن يكون الملف الأصلى مفروزا (Sorted) أو مفهرسا (Indexed) على الحقل المفتاح (Key field) .

الشكل العام:

TOTAL ON<key field> TO <filename>[FIELDS <field list>]
[<scope>][FOR <cond>][WHILE <cond>]

فى حالة عدم تحديد شروط باستخدام FOR <cond> أو WHILE <cond> فسوف يتم جمع الحقول الرقمية فى كل السجلات .

مثال :

```
. USE PAYROLL ORDER JOBCODE
Master index: JOBCODE
. DISPLAY ALL
Record #        NO NAME        JOPCODE     SALARY
      1          1 AHMED          11        200.00
      3          3 MOHAMED        11        230.00
      2          2 ESLAM          12        211.00
      5          6 EMAN           12        321.00
      6          7 ADEL           12        234.00
      4          4 HESHAM         13        330.00
      7          9 TAREK          13        400.00
. TOTAL ON JOBCODE TO SAL FIELD SALARY
      7 records totaled
      3 records generated
. USE SAL
. DISPLAY ALL
Record #        NO NAME        JOPCODE     SALARY
      1          1 AHMED          11        430.00
      2          2 ESLAM          12        766.00
      3          4 HESHAM         13        730.00
```

فى هذا المثال يتم جمع حقل (Salary) للسجلات التى تنتمى لكل Jobcode .

إعداد المتوسط الحاسبى (Average)

لحساب المتوسط الحسابى لحقل رقمى أو مجموعة حقول رقمية ...

الشكل العام:

```
AVERAGE[<expN list >[<scope>] FOR<cond>]
[WHILE <cond>][TO<memvar list >/TO ARRAY<array name>]
```

مثال :

```
. USE PAYROLL ORDER JOBCODE
Master index:  JOBCODE
. DISPLAY ALL
Record #          NO NAME           JOPCODE        SALARY
      1            1 AHMED              11         200.00
      3            3 MOHAMED            11         230.00
      2            2 ESLAM              12         211.00
      5            6 EMAN               12         321.00
      6            7 ADEL               12         234.00
      4            4 HESHAM             13         330.00
      7            9 TAREK              13         400.00
. AVERAGE SALARY FOR JOBCODE = 12
      3 records averaged
      SALARY
      255.33
. AVERAGE SALARY FOR JOBCODE = 13
      2 records averaged
      SALARY
      365
. AVERAGE SALARY FOR JOBCODE = 11
      2 records averaged
      SALARY
      215
```

فى هذا المثال يتم إعداد المتوسط الحسابى للمرتب (Salary) لكل وظيفة (Jobcode)

الأمر CALCULATE

لإنجاز أكثر من وظيفة مالية (Financial) أو إحصائية (Statistical) خلال سطر أمر واحد ، وهذا يتيح سرعة أكبر من إنجاز نفس الأمر فيما لو تم انجاز كل وظيفة على حدة ؛ وذلك لأن تنفيذ هذه الوظائف بواسطة الأمر CALCULATE يكتفى بالمرور على سجلات ملف قاعدة البيانات مرة واحدة.

الشكل العام:

CALCULATE[scope]<options list>[FOR<cond>][WHILE<cond>] TO <memvar lists/TO ARRAY <array name>]

والوظائف المتاحة فى <options list > **هى :**

أ - الوظائف المالية [Financial functions]

NPV (<rate>, <flows>, <inital>) (net present value)

ب - الوظائف الإحصائية [Statistical functions]

١- AVG(expN>) لحساب المتوسط الحسابى.

٢- CNT ( ) لعد السجلات فى ملف قاعدة البيانات .

٣- MAX (<exp>) لإيجاد أكبر رقم فى حقل معين فى قاعدة البيانات .

٤- MIN(<exp>) لإيجاد أصغر رقم فى حقل معين فى ملف قاعدة البيانات .

٥- STD (<expN>) لحساب حيود (Deviation) القيم .

٦- SUM (<expN>) لجمع قيم حقل معين فى ملف قاعدة البيانات .

مثال :

```
. USE PAYROLL ORDER JOBCODE
Master index:  JOBCODE
. DISPLAY ALL
Record #          NO NAME        JOPCODE      SALARY
      1            1 AHMED            11      200.00
      3            3 MOHAMED          11      230.00
      2            2 ESLAM            12      211.00
      5            6 EMAN             12      321.00
      6            7 ADEL             12      234.00
      4            4 HESHAM           13      330.00
      7            9 TAREK            13      400.00
. CALCULATE AVG (SALARY), SUM (SALARY) FOR JOBCODE = 11
      2 records
      AVG (SALARY)   SUM (SALARY)
               215            430
```

```
. CALCULATE AVG (SALARY), SUM (SALARY) FOR JOBCODE = 12
        3 records
        AVG (SALARY)   SUM (SALARY)
              255.33             766
. CALCULATE AVG (SALARY), SUM (SALARY) FOR JOBCODE = 13
        2 records
        AVG (SALARY)   SUM (SALARY)
                365              730
```

فى المثال السابق يتم حساب المتوسط الحسابى لحقل (Salary) وكذلك حاصل جمع حقل Salary لكل وظيفة على حدة.

الفصل التاسع

ترتيب البيانات

* طرق ترتيب السجلات.
* الفرز.
* الفهرسة.
* البحث عن البيانات.

الفصل التاسع

ترتيب البيانات

هناك طريقتان لترتيب السجلات هما:

١- الفرز SORTING .
٢- الفهرسة INDEXING .

الفرز SORTING

ينشئ ملف قاعدة بيانات جديد يحتوى على بيانات ملف قاعدة البيانات الأصلى أو بعضاً منها مرتبة ترتيبا معينا ، إما تصاعديا أو تنازليا.

الصورة العامة :

SORT TO <filen ame>ON<field1>[/A][/C][/D]
[,<field2>[/A][/B][/C] ...][ASCENDING]/EDSC.]
[<scope>][FOR<condition>][WHILE<condition>]

١- A/ تعنى ترتيبا تصاعديا Ascending .
D/ تعنى ترتيبا تنازليا Descending .
C/ تعنى أن الترتيب لا يفرق بين الحروف الكبيرة (Uppercase) والصغيرة (Lowercase) .

٢- الأمر SORT يتيح إنشاء ملف مرتباً ترتيباً معيناً ، إما تصاعديا أو تنازليا ، يحتوى فقط على السجلات التى تقابل شرط ما وذلك باستخدام FOR <cond> , WHILE <cond> .

٣- يمكنك استخدام حتى عشرة حقول كحد أقصى للفرز.

مثال :

```
USE STORE
SORT ON ITEMNO/A, BALANCE TO STORSRT
```

فى هذا المثال يتم فرز الملف على الحقلين (Itemno) و (Balance) ويكون ترتيب أولهما تصاعديا ، وإسم الملف المفروز (Storsrt) .

الفهرسة INDEXING

ينشئ فهرسا يكون مرتبطا بملف قاعدة البيانات ويحتوى على رقم السجل (Record number) ، والحقل المفتاح (Key field) مرتباً ترتيباً عددياً أو هجائيا ، وتكون الفهرسة مرتبة تصاعدياً ما لم تحدد الإختيار « تنازليا » (Descending)

الشكل العام :

```
INDEX ON < key exp > TO < ndx. filename > / TAG < tag name >
[ OF < .ndx filename > ] [ uniq ue ] [ Desc. ]
```

أولا :

```
INDEX ON < key exp > TO < .ndx filename > [ unique ] [ Desc.]
```

ينتج فهرساً من النوع (.ndx) يحتوى على الحقل المفتاح (key field) ورقم السجل المناظر له .

وهذا النوع من الفهرسة هو المستخدم فى فهرسة الملفات فى dBase II و dBase III+

وهذا النوع يستلزم فتح ملف فهرسة لكل فهرسة على ملف قاعدة البيانات .

فمثلاً إذا أردنا فهرسة الملف (Store .dbf) على الحقل (Item no) فإن ذلك يستلزم إنشاء ملف فهرسة يحتوى على الحقل المفتاح ، ورقم السجل (Record num ber) المناظر له .

وإذا أردنا فهرسة نفس الملف مرة أخرى على الحقل (Item name) ، فإن ذلك يستلزم إنشاء ملف فهرسة آخر يحتوى على الحقل المفتاح ورقم السجل المناظر له .

وفى هذه الحالة يكون لدينا ثلاثة ملفات هى ملف قاعدة البيانات الأصلى (Store) ، وملف الفهرسة الأول ، وملف الفهرسة الثانى .
UNIQUE تعنى التغاضى عن السجلات المكررة فى الفهرسة.

ثانيا :

INDEX ON <key exp> TAG <tagname>[OF <.mdx filename>] [unique][Desc.]

هذا النوع من الفهرسة تتيحه dBase IV لأول مرة ، وهو يتيح استخدام حتى ٤٧ فهرسا (Tags) تكافئ ٤٧ ملف فهرسة من النوع السابق (.ndx) فى ملف واحد يسمى ملف الفهرسة المتعددة (.mdx) - (Multible index) .
وعند إنشاء أو تعديل ملف قاعدة البيانات يكون لديك الإختيار لتحديد حقول الفهرسة (حتى ٤٧ حقلا) ، وإذا تم اختيار أى حقل كحقل فهرسة ، فإنه ينشئ ملف (.mdx) بنفس اسم ملف قاعدة البيانات .
والملف (.mdx) يفتح تلقائيا بمجرد فتح ملف قاعدة البابينات المرتبطة به ، ويتيح أيضا إمكانية الفهرسة التنازلية (Desc.) .

مثال :

```
USE STORE
INDEX ON ITEMNO TO NUMINDEX
INDEX ON ITEMNAME TO NAMEINDEX
```

فى هذا المثال تمت الفهرسة من النوع (.ndx) وقد تمت الفهرسة على حقلين هما : (itemno) و (itemname) - لاحظ أن ملفات قاعدة البيانات والفهارس تستلزم فتح ٣ ملفات .

لاحظ المثال التالى :

```
USE STORE
INDEX ON ITEMNO TAG NUMBER
INDEX ON ITEMNAME TAG NAME
```

فى هذا المثال ملفان فقط مطلوب فتحهما هما : ملف قاعدة البيانات (Store) وملف الفهرسة المتعددة (Store .mdx) .

فتح ملفات الفهرسة

الأمر SET INDEX يتيح فتح ملفات مفهرسة محددة من النوع (.ndx) أو النوع (.mdx) ، ومن الممكن أن يحدد الفهرس المتحكم للملف قاعدة البيانات .

الشكل العام:

```
SET INDEX TO [<filename list>[ORDER [TAG]
<ndx filename>/<tagname>[OF<.mdx filename>]]]]
```

مثال :

```
USE STUDENT
SET INDEX TO STUDNET.MDX ORDER RESULTS
MASTER INDEX:  RESULTS
```

فى هذا المثال يتم فتح ملف الفهرسة المتعددة (Student .mdx) وتحديد الفهرس المتحكم (Controlling index) وهو فهرس Result .

تغيير الفهرس المتحكم

فى ملف الفهرسة المتعددة (.mdx) يوجد أكثر من فهرس ، ولكن فهرسا واحدا فقط هو الذى يتحكم فى فهرسة ملف قاعدة البيانات فى نفس الوقت . ولتغيير هذا الفهرس استخدم الأمر SET ORDER TO <tagname> .

مثال :

```
USE STUDENT
SET INDEX TO STUDENT.MDX ORDER RESULTS
MASTER INDEX: RESULTS
SET ORDER TO STUDENTNO
MASTER INDEX: STUDENTNO
? ORDER ( )
STUDENTNO
```

فتح ملف قاعدة البيانات مع تحديد الفهرس المتحكم فى نفس الوقت

إذا أردت مثلا فتح الملف (Student) وتخصيص المنطقة العاملة فى الذاكرة رقم ٢ ، وتحديد الفهرس Results كفهرس تحكم ؛ إكتب ...

```
USE STUDENT IN 2 ORDER RESULTS
```

البحث عن البيانات SEARCHING

هناك طريقتان للبحث عن البيانات هما :

١- الطريقة التتابعية (Sequential Search)

وتستخدم الأمر LOCATE أو الوظيفة LOOKUP ( ) ويستخدمان فى الملفات غير المفهرسة .

٢- طريقة البحث العشوائى (Random Search)

تستخدم الأمر FIND والأمر SEEK ؛ كذلك يمكن استخدام الوظيفة LOOKUP ( ) والوظيفة SEEK ( ) .

الأمر LOCATE

يبحث عن سجل فى ملف قاعدة بيانات يقابل شرط معين .

الصورة العامة:

```
LOCATE [FOR]<cond>[<scope>][WHILE <cond>]
```

مثال :

```
USE STORE
DO WHILE .T.
IF .NOT. FOUND ( )
EXIT
ENDIF
LOCATE FOR ITEMNAME = "CABLES" WHILE ITEMNO <1000
? ITEMNO, ITEMNAME, RECORDLVL
CONTINUE
IF .NOT. FOUND ( )
EXIT
ENDIF
ENDDO
```

لاحظ وجود الأمر (CONTINUE) الذى يواصل البحث عن السجل التالى فى الملف الذى يحقق نفس الشرط.

الأمر SEEK

يبحث عن أول سجل فى ملف قاعدة البيانات المفهرس يقابل تعبيرا معينا.

الشكل العام:

```
Seek <exp>
```

مثال :

```
USE STUDENT ORDER STUDENTNO
DO WHILE .T.
MNO = O
@ 5,5 SAY "PLEASE ENTER THE STUDENT NUMBER:" GET MNO
SEEK MNO
IF .NOT. FOUND ( )
? "THIS NUMBER NOT EXIST.. PLEASE RE-ENTER.."
WAIT ""
LOOP
ENDIF
EXIT
ENDDO
```

لاحظ أن الملف Student مفهرس على الحقل المفتاح (Studentno) لذلك فإن التعبير (Expression) الذى يستخدمه الأمر Seek فى البحث هو رقم يمثل الحقل (Studentno) .

الوظيفة FOUND ( )

تعطى إجابة منطقية من حرف واحد ؛ إما .T. أو .F. ، فى حالة تحقق أو عدم تحقق شرط البحث عن السجلات (Records) عند استخدام الأوامر ... SEEK, LOCATE, FIND, CONTINUE

الوظيفة SEEK ( )

تبحث فى الملف المفهرس عن سجل محدد وتعطى نتيجة إما إيجابية .T. أو سلبية .F.

الصورة العامة : SEEK(<exp>[,<alias>])

alias تشير لرقم منطقة الذاكرة (Work area).

وتكافئ هذه الوظيفة استخدام كل من الأمر SEEK والوظيفة FOUND ( ) معا .

مثال :

```
SEEK MNO
IF FOUND ( )
DO PRG1
ENDIF
```

يمكن عمل المثال السابق بالطريقة الآتية باستخدام الوظيفة SEEK ( ) :

```
IF SEEK (MNO)
DO PRG1
ENDIF
```

الأمر FIND

يبحث عن أول سجل فى ملف قاعدة البيانات يقابل الرقم أو التعبير الحرفى المطلوب .

الصورة العامة: FIND <literal key> .

مثال :

```
USE STUDENT ORDER NAME
MNAME = SPACE (10)
@ 5,5 SAY "PLEASE ENTER THE NAME:" GET MNAME
FIND &MNAME
IF FOUND ( )
@ 8,5 SAY NAME
@ 10,5 SAY CLASS
ENDIF
```

الفصل العاشر

التعامل مع قواعد البيانات

* التعامل مع أكثر من ملف فى نفس الوقت.
* ربط الملفات.
* دمج الملفات.
* إستخدام الإستفسارات فى البرنامج.

الفصل العاشر

التعامل مع قواعد البيانات

التعامل مع أكثر من ملف فى نفس الوقت

عند فتح ملف قاعدة بيانات ما لإجراء أى عمليات على أحد السجلات (Record) ، ثم قمت بفتح ملف آخر لإجراء عملية ما على أحد سجلاته ، وعند إعادة فتح الملف الأول ستكتشف أن مؤشر السجلات (Pointer) لم يعد عند نفس السجل الذى سبق وأن كنت فيه ، وستجد أن المؤشر قد انتقل الى السجل الأول فى الملف.

ولكننا نحتاج أحيانا لأن نتعامل مع أكثر من ملف واحد مفتوح فى نفس الوقت وذلك بغرض ربط أكثر من ملف لإنشاء قواعد بيانات لها علاقات مشتركة ، وكذلك للعمل مع أكثر من ملف فى نفس الوقت لإجراء العمليات عليهم فى سرعة .

لذلك فإنه أمكن تقسيم الذاكرة لعشرة مناطق (من ١ الى ١٠) أو (من A الى J) أو بإسم رمزى (ELIAS) . وكل منطقة يمكن أن تحتوى على ملف قاعدة بيانات ، ويتم تخصيص أى منطقة من هذه المناطق لملف قاعدة بيانات باستخدام الأمر SELECT ولا تستطيع أن تتعامل فى لحظة ما إلا مع منطقة واحدة تسمى المنطقة الإبتدائية (Primary) وباقى المناطق تكون فى المنطقة الثانوية (Secondary) .

فإذا كنت تتعامل مع ملف فى أحد المناطق ، ولتكن المنطقة A ، وأردت أن تتعامل مع الملف الموجود بالمنطقة B ، فإنك تستطيع ذلك باستخدام الأمر SELECT B ، وسيحل الملف الجديد فى المنطقة B محل الملف فى المنطقة A ، ولكن مؤشر السجلات (Pointer) فى الملف الموجود فى المنطقة A لا يتغير ، أى يظل عند آخر سجل كان يقف عنده .

الأمر SELECT

يتيح اختيار منطقة ذاكرة عاملة (Memory work area) او تخصيص منطقة ذاكرة لملف مفتوح .

مثال :

```
SELECT 1
USE FILE1
SELECT2
USE FILE2
```

فى هذا المثال يتم تخصيص المنطقة ١ للملف File1 ، والمنطقة ٢ للملف File2 .

ملحوظة : يمكنك تحديد منطقة الذاكرة فى نفس الوقت الذى تفتح فيه الملف الذى تحتويه هذه المنطقة ، وذلك باستخدام الأمر

```
USE <filename> IN<n>
```

مثال :

```
USE FILE1 IN2
```

فى هذا المثال يتم تخصيص المنطقة ٢ للملف File1 مع اختيار المنطقة ٢ للعمل (أى يكون الملف File1 مفتوحا).

ربط الملفات

يمكنك ربط ملفات قواعد البيانات تبعا لمفتاح الفهرسة (Key) أو رقم السجل (Record Number) ، وذلك باستخدام الأمر SET RELATION .

مثال :

```
SELECT1
USE STORE ORDER ITEMNO
SELECT2
USE STORE1
SET RELATION TO ITEMNO INTO STORE LIST NEXT 10;
ITEMNO, DISCRIP, STORE --> PRICE
```

فى هذا المثال ...

الملف Store1 يسمى الملف الأب ، والملف Store يسمى الملف الإبن ، ويُربط الملف الأول بالثانى ، وعند الوصول الى سجل ما فى الملف الأب

فإنه يتم تلقائيا إيجاد أول سجل فى الملف الإبن يتوافق مع سجل الملف الأب فى المفتاح (Itemno). لاحظ أن الملف الإبن (Store) يجب أن يكون مفهرسا على المفتاح ، وعند تحرك مؤشر السجلات فى الملف الأب يتحرك بالتالى مؤشر السجلات (Itemno) فى الملف الإبن للسجل الذى يتوافق فى المفتاح مع السجل فى الملف الأب .

يلاحظ فى المثال السابق أن مؤشر السجلات (Pointer) فى الملف الإبن سوف يقف عند أول سجل فقط مرتبط بالسجل فى الملف الأب ومتوافق فى المفتاح.

وبفرض أنه يوجد سجلات أخرى فى الملف الإبن متوافقة فى المفتاح ، فللوصول الى هذه السجلات قبل أى حركة للمؤشر فى الملف الأب ، يستخدم الأمر SET SKIP TO [ALIAS1[,ALIAS2]..]

ملحوظة : فى حالة عدم وجود سجل متوافق فى الملف الإبن ، فإن مؤشر السجلات يوضع فى نهاية الملف (EOF).

مثال :

```
SELECT2
USE MON ORDER NO
USE STORE IN 1 ORDER NO
SET RELATION TO 6 --> NO INTO A
SET FIELDS TO A --> NO, A --> NAME, b --> R1,; b--> R2
? NAME
? NO
```

فى هذا المثال يكون الملف (Store) هو الملف الأب ، ويكون الملف (Mon) هو الإبن . يتم ربط الملفين بالحقل المفتاح NO ؛ لاحظ أن الملف الإبن مفهرسا على المفتاح NO .

ربط الملف بأكثر من ملف

يمكن ربط الملف بأكثر من ملف واحد والتعامل معهم فى نفس الوقت ، وفى هذه الحالة يوجد أكثر من ملف إبن ، ومن الممكن أن يكونوا مفهرسين على أكثر من مفتاح مختلف.

مثال :

```
SELECT1
USE STORE1 ORDER ITEMNO
SELECT2
USE STORE
SELECT3
USE STORE2 ORDER TYPE
SELECT2
SET RELATION TO ITMENO INTO STORE1, TYPE INTO STORE2;
LIST NEXT10 ITEMNO,  STORE1 --> PRICE, STORE2 --> SUPPL.
```

لاحظ أن الملفين الإبن Store1, Store2 مفهرسين على مفتاحين مختلفين.

دمج الملفات

من الممكن دمج حقول وسجلات محددة من ملفي قاعدة بيانات مفتوحين ، وإنشاء ملف جديد باستخدام الأمر (JOIN) .

الشكل العام:

JOIN WITH <alias> TO<filename> FOR <cond>[FIELD<fields list>]

١- تشير Alias للإسم الرمزى للملف المفتوح فى منطقة ثانوية .

٢- يوضع مؤشر السجلات (Pointer) فى أول سجل فى الملف فى المنطقة المفتوحة (Primary) والمؤشر فى الملف فى المنطقة الثانوية (Secondary) يبحث عن السجل الذى يحقق الشرط (FOR<cond>) ، فإذا تحقق فإن سجلا جديدا يضاف الى الملف الجديد .

٣- لاحظ أن المؤشر فى المنطقة المفتوحة لا يتحرك الى السجل التالى إلا بعد المرور على جميع السجلات فى الملف فى المنطقة الثانوية .

مثال :

```
USE STORE1
USE STORE2 IN2
JOIN WITH STORE1 TO STORE3 FOR ITEMNO = B -->ITEMNO;
FIELDS ITEMNO, ITEMNAME, B --> BALANCE, B --> TBAL
```

فى هذا المثال يتم دمج سجلات من كل من الملف Store1 ، والملف Store2 اللذين يحققان الشرط وهو اتفاق الملفين فى قيمة حقل (Itemno) الى ملف جديد هو Store3 الذى يحتوى على الحقول الآتية Itemname, Itemno من المنطقة A ، وكل من TBal, Balance من المنطقة B .

تحديث الملفات UPDATING

من الممكن استبدال محتويات حقول فى ملف مفتوح ببيانات من ملف آخر باستخدام الأمر UPDATE علي أن تكون سجلات الملفين متوافقة على مفتاح واحد (Key Field) .

الصورة العامة:

```
UPDATE 0N <key field> FROM <alias> REPLACE <field1 name1>
WITH <exp1>[,<field name2> WITH <exp2> ..][RANDOM]
```

مثال :

```
USE TRANSACT ORDER ITEMNO
USE MASTER ORDER ITEMNO IN 2
UPDATE ON ITEMNO FROM TRANSACT REPLACE
TBALANCE WITH;
TRANSACT --> BAL1 + TRANSACT --> BAL2
```

فى هذا المثال يتم تحديث الملف (Master) باستبدال محتويات الحقل (TBalance) بحاصل جمع محتويات الحقلين Bal2, Bal1 من الملف (Transact) .

إستخدام الإستفسارات (Queries) فى البرنامج

الإستفسارات نوعين ...

١- إستفسارات العرض VIEW QUERIES
٢- إستفسارات التحديث UPDATE QUERIES

يمكن دمج الإستفسارات فى البرنامج بواحدة من أربع طرق مختلف :

١- يتم أولا إنشاء ملف الإستفسار بواسطة الأمر CREATE/MODIFY QUERY/VIEW الذى ينقلك الى شاشة تصميم الإستفسارات (Queries Design Screen) ، ثم اجعل البرنامج يحتوى على الأمر

SET VIEW TO <.qbe file>

٢- إنشاء ملف استفسار متوافق مع dBase III+ باستخدام الأمر

CREATE VIEW FROM INVIRONMENTS

ثم اجعل البرنامج يحتوى على الأمر

SET VIEW TO <.VUE filename>

٣- إنشاء ملف استفسار بواسطة الأمر CREATE/MODIFY QUERY/VIEW ثم انسخ (Copy) الأوامر من الملف (.qbe) الى ملف (.prg) ، وبذلك تستطيع التعديل فى ملف البرنامج بواسطة الأمر

MODIFY COMMAND <.prg filename>

٤- بكتابة الأوامر مباشرة فى البرنامج

١- VIEW QUERIES

يختار سجلات معينة من ملف قاعدة البيانات ويعرض حقول محددة منها ، ويدمج بيانات من أكثر من ملف قاعدة بيانات ، وهى توضع فى الذاكرة الرئيسية (RAM) ، ولذلك يكون الأداء أسرع ، وعند استخدامها لإمداد التقارير بالبيانات فإنها تعطى إمكانية للتحكم فى أى السجلات سوف تطبع.

أمثلة عملية لإنشاء برامج عرض استفسار ... الملف المستخدم فى الأمثلة هو (Student) وبنية الملف مبينة فى الشكل التالى .

1	STUDENTNO	Numeric	3	0	Y
2	STUDNAME	Character	30		N
3	CLASS	Numeric	2	0	N
4	CHEMISTRY	Numeric	5	2	N
5	PHYSICS	Numeric	5	2	N
6	MATH	Numeric	5	2	N

Database/C:/dbase/STUDENT /Field 1/6 Caps
Position selection bar: Select: Leave menu: Esc
Save any changes made and leave database design

مثال - ١:

إنشاء ملف عرض إستفسار لعرض حقول محددة من سجلات محددة تحقق شرط Student <20 .AND. class = 1

```
1 SET FIELDS TO
2 SELECT1
3 USE STUDENT
4 SET EXACT ON
5 SET FILTER TO ((A->STUDENTNO<20 .AND. A-> CLASS = 1))
6 SET FIELDS TO A->STUDENTNO,A- CLASS,A->CHEMISTRY,
  A->;
7 PHYSICS,A->MATH,;
8 RESULT = ((A->MATH+A->PHYSICS+A->CHEMISTRY)/3),A->;
9 STUDNAME
10 RETURN
```

السطر الخامس

```
SET FILTER TO ((A -> STUDENTNO <20 .AND. A -> CLASS = 1))
```

يعنى ترشيح سجلات معينة من الملف تحقق الشرطين السابقين

السطر السادس

SET FIELDS TO A -> STUDENTNO ...

تعنى تحديد الحقول الموضحة فى السطر ليتم تنفيذ الأوامر التالية عليها فقط دون بقية حقول الملف .

مثال -٢:

إنشاء ملف عرض استفسار (View) لاختيار السجلات التى تحقق الشرط STUDENTNO > 50 .OR. CLASS = 2 وتحديد حقول معينة من هذه السجلات وهى حقول (Studname, Class, Studentno) .

```
SET FIELDS TO
SELECT1
USE STUDENT
SET EXACT ON
SET FILTER TO ((A->STUDENTNO>50).OR.(A->CLASS=2))
SET FIELDS TO A->STUDENTNO,A->CLASS,A->STUDNAME
```

مثال -٣:

لإنشاء ملف عرض استفسار بدمج سجلات من الملفين Student و Student1 ويحدد الحقول أدناه لينفذ عليها أوامر معينة :

١- حقل Studname من الملف الأول .
٢- حقل Class من الملف الأول
٣- حقل Type من الملف الثانى
٤- حقل Nation من الملف الثانى
٥- حقل Studentno من الملف الثانى

```
SET FIELDS TO
SELECT 2
USE STUDENT1
USE STUDENT IN 1 ORDER STUDENTNO
SET EXACT ON
SET FILTER TO
SET RELATION TO B -> STUDENTNO INTO A
SET SKIP TO A
GO TOP
SET FIELDS TO A->STUDNAME,A->CLASS,B->TYPE,B->;
NATION,B->STUDENTNO
```

٢- UPDATE QUERIES

تتيح تغيير حقول وسجلات مختارة فى ملف قاعدة البيانات ، وكل Update Query يعدل فى ملف قاعدة بيانات محدد ، وبذلك يتيح لك إمكانية كبيرة للتعديل فى البيانات بشكل سريع وسهل .

مثال -٤:

إنشاء ملف استفسار تحديث يقوم بتعديل درجات الطلاب الراسبون فى مواد الكيمياء والطبيعة والرياضيات ، وذلك بإضافة درجتين لكل طالب فى ملف Student.

```
SET FIELDS TO
SELECT1
USE STUDENT
SET EXACT ON
SET FILTER TO ((A->CHEMISTRY<50.AND.A->PHYSICS<50.AND.A->MATH<50))
REPLACE ALL CHEMISTRY WITH A->CHEMISTRY+2,PHYSICS WITH A->;
PHYSICS+2,MATH;WITH A->MATH+2
SET FILTER TO
SET FIELDS TO
SET RELATION TO
```

مثال -٥:

إنشاء ملف استفسار تحديث لإضافة سجلات معينة من الملف Student التى تحقق شرط Studentno >50 الى الملف Test

```
SELECT 2
USE TEST
SET FIELDS TO
SELECT 1
USE STUDENT
SET FILTER TO ((A->STUDENTNO>50))
DO WHILE .NOT. EOF ( )
SELECT 2
APPEND BLANK
SELECT 1
SKIP
ENDDO
SET FILTER TO
SET FIELDS TO
SET RELATION TO
USE
```

مثال -٦:

لإنشاء ملف استفسار تحديث لحذف (Delete) السجلات من ملف Student التى تحقق شرط Studentno > 44 .

```
SET FIELDS TO
SELECT 2
USE STUDENT
SET FILTER TO ((B->STUDENTNO >44))
DELETE ALL
SET FILTER TO
SET FIELDS TO
SET RELATION TO
```

الفصل الحادى عشر

التقارير

* العرض السريع للبيانات.
* إنشاء التقارير المنسّقة.
* مكونات برنامج التقرير.
* متغيرات ذاكرة النظام.

الفصل الحادى عشر

التقارير REPORTS

التقرير هو تنظيم البيانات فى صورة منسقة مرسلة الى الشاشة أو الى الطابعة . كذلك يمكن عرض البيانات أو إرسالها للطابعة بإسلوب سريع ولكنه غير مجهز وغير منسق . وفيما يلى شرح لبعض الأوامر المستخدمة لتحقيق هذا الغرض.

أولا : العرض السريع للبيانات

١- الأمر LIST/DISPLAY

يعرض محتويات ملف قاعدة بيانات فى شكل غير مجهز وعلى هيئة أعمدة .

الصورة العامة:

LIST/DISPLAY [[FIELDS]<expressions list>][OFF]
[<scap>][FOR<cond>][WHILE<cond>]
[TO PRINTER/TO FILE <filename>]

أ – While, For يحددان الشروط لعرض السجلات التى تحققها فقط ، وإذا لم يكتب While, For ضمن الأمر ، فإنه يتم عرض كل سجلات الملف.

ب – الفرق بين Display, List هو أنه فى حالة استخدام Display يتوقف العرض بعد كل شاشة بينما فى حالة استخدام List لا يتوقف العرض حتى آخر سجل . ويمكن ايقاف العرض باستخدام Display أو List بضغط Ctrl-S ، ويمكن الغاء العرض بضغط ESC.

٢- الأمر ?/??

يعرض أو يطبع تعبير أو أكثر ، أو رسالة قصيرة أو حقل أو أكثر ، أو يعرض نتائج العمليات الحسابية .

الصورة العامة:

```
? /?? [<exp1>[PICTURE<exp2>]
[FUNCTION<function list>][AT<expN>]
[STYLE<font number>]]
[,<exp2>...][,]
```

? تعرض أو تطبع فى أول السطر التالى لاحتوائها على Carriage Return (R.C) و Linefeed (L.F) .

?? تعرض أو تطبع بداية من المكان الحالى للمؤشر أو الموقع الحالى للطبع .

Function, Picture لتحديد شكل البيانات المعروضة أو المطبوعة باستخدام دوال التشكيل (Format Function) أو رموز قوالب (Templete Symbol).

Style لتحديد طراز الطباعة مثل الخط المائل (Italic) أو وضع خط تحت الكتابة (Underlined).

٣- الأمر TEXT ... ENDTEXT

يرسل نصا (Text) للعرض على الشاشة أو للطابعة .

الصورة العامة:

```
TEXT
<text characters>
ENDTEXT
```

٤- الأمر ... SAY (R,C) @

يعرض تعبير معين أو حقل فى المكان المحدد بالإحداثيان (R, C).

ثانيا: إنشاء التقارير المنسقة/المجهزة

يمكنك إنشاء التقرير المنسق بإحدى طريقتين:
أ - باستخدام شاشة تصميم التقارير (إنظر فصل التقارير فى الجزء الأول).
ب - إنشاء برنامج التقرير باستخدام أوامر dBase IV

مكونات برنامج التقرير

يتكون برنامج التقرير من برنامج رئيسى (Main Program) ، ومجموعة من الإجراءات (Procedures) أو البرامج الفرعية (Subprograms) .

البرنامج الرئيسى يتم فيه الآتى :

١- التحقق من حالة الطابعة (Printer) أى أن تكون الطابعة فى الوضع ON ويكون الإتصال جيدا بينها وبين الحاسب .
٢- تنفيذ الإجراء الخاص باستقبال أوضاع الطابعة المناسبة من المستخدم .
٣- إتاحة إمكانية وقف الطبع عند الضغط على مفتاح ESC .
٤- التحكم فى كمية البيانات المطبوعة فى الصفحة الواحدة بحيث لا تطبع بيانات صفحة على الصفحة التالية .
٥- تنفيذ إجراء رأس الصفحة (Page Head) وإجراء تذييل الصفحة (Page Footer) ، وإجراء تفاصيل التقرير (Details) .
٦- إنشاء الحلقة الدوارة (Do While) لطبع بيانات محددة أو كل بيانات ملف قاعدة البيانات .
٧- إعادة ضبط الوضع المسبق لمتغيرات ذاكرة النظام (System memvars) الخاصة بأوضاع الطباعة .

الأوامر الهامة المستخدمة فى برنامج التقرير

١- الأمر PRINTJOB; ENDPRINTJOB

ينفذ مهام الطباعة المرتبطة بالأوضاع التى تحدد باستخدام متغيرات ذاكرة النظام (System Memory Variables) .

الشكل العام:

```
PRINTJOB
<commands>
ENDPRINTJOB
```

PRINTJOB ينفذ التالى :

أ - ينفذ شفرة بداية الطباعة (_Pscode)

ب - يدفع الورق صفحة للأمام إذا كان (_Peject) مضبوطا على الوضع (Before) أو (Both) .

جـ - يحدد القيمة (صفر) للمتغير (_Pcolno)

ENDPRINTJOB ينفذ التالى :

أ - ينفذ شفرة نهاية الطباعة (_Pecode) .

ب - يدفع الورق صفحة إذا كان (_Peject) مضبوطا على الوضع (After) أو (Both) .

جـ - ينفذ عدد مرات نسخ التقرير المحددة بالمتغير (_Pcopies)

الأمر ON PAGE

الصورة العامة :

```
ON PAGE [AT LINE<expN><command>]
```

أ - يتيح تنفيذ إجراء (Procedure) تذييل الصفحة (Footer) عند نهاية كل صفحة . لإيجاد قيمة <expN> استخدم الصيغة ...

<expN> = page length - bottom margin - footer height

ب - يجب التأكد من أن عدد أسطر التذييل لا تزيد عن عدد الأسطر المتبقية من الصفحة ، وإلا كتب التذييل فى مقدمة الصفحة التالية .

مثال :

```
ON PAGE AT LINE -PLENGTH-2 DO FOOT
PROCEDURE FOOT
?
? "To be continue" AT 40
EJECT PAGE
RETURN
```

نموذج البرنامج الرئيسى فى برنامج التقرير

```
1  SET TALK OFF
2  SET ECHO OFF
3  DO PRSTAT
4  DO ENVIR
5  CONTNU = .T.
6  ON ESCAPE DE REPEX
7  IF _PSPACING >1
8  CLINE = _PLENGTH -(_PSPACING+1)
9  ELSE
10 CLINE = _PLENGTH -2
11 ENDIF
12 PRINTJOB
13 ON PAGE AT LINE CLINE DO FOOT
14 DO HEAD
15 DO WHILE .NOT. EOF( ) .AND. CONTNU
16 DO DETAILS
17 ENDDO
18 ENDPRINTJOB
19 DO REINVIR
20 RETURN
```

شرح البرنامج الرئيسى

١- السطر (٣) لتنفيذ الإجراء (Prstat) الخاص بالتحقق من حالة الطابعة قبل عملية الطباعة .

٢- السطر (٤) لتنفيذ الإجراء (Envir) الخاص باستقبال أوضاع الطباعة المناسبة للمستخدم .

٣- السطر (٥) لإنشاء حقل ذاكرة متغير منطقى (Contnu) وضبطه على الوضع (.T.) ، ويستخدم فى الحلقة الدوارة (من سطر ١٥ حتى سطر ١٧) كشرط لاستمرار طباعة بيانات الملف ، ويمكن تغيير الوضع (.T.) الى الوضع (.F.) وذلك بضغط مفتاح ESC الذى ينفذ الإجراء (Repex) - سطر ٦.

٤- من السطر (٨) الى (١١) حلقة شرطية لضبط موقع تذييل الصفحة فى حالة اختيار المستخدم لأى اختيار للمسافة بين السطور ، وهى إما ١ أو ٢ أو ٣.

٥- السطر (١٢) لتنفيذ مهام الطباعة المرتبطة بالأوضاع التى تم تحديدها باستخدام متغيرات النظام (System memvars) المنفذة باستخدام الإجراء (Envir) فى السطر (٤).

٦- السطر (١٣) لتنفيذ الإجراء (Foot) عند نهاية كل صفحة.

٧- السطر (١٤) خاص بتنفيذ إجراء رأس الصفحة (Head) .

٨- من السطر (١٥) وحتى (١٧) حلقة دوارة للتحكم فى طباعة بيانات ملف قاعدة البيانات مادام مؤشر السجلات داخل الملف ، ولم يضغط المستخدم مفتاح ESC الذى يقوم بالتالى بتغيير حالة (Contnu) من الوضع (.T.) الى الوضع (.F.) .

٩- السطر (١٨) لإنهاء عملية الطباعة أو إعادة طبع نفس التقرير عدد المرات المحددة فى المتغير (_Pcopies) ، وكذلك دفع الورق صفحة للأمام فى حالة أن يكون المتغير (_Peject) فى الوضع (Both) أو فى الوضع (After) .

١٠- السطر (١٩) لتنفيذ الإجراء (Reenvir) الخاص بإعادة ضبط متغيرات ذاكرة النظام للوضع المسبق .

نموذج لإجراء التحقق من حالة الطابعة قبل عملية الطباعة

```
PROCEDURE PRSTAT
1  SET TALK OFF
2  SET ECHO OFF
3  SET ESCAPE OFF
4  DEFINE WINDOW REP1 FROM 10,10TO 15,70
5  KEY = 0
6  ACTIVATE WINDOW REP1
7  ? "Please ready the printer or press (ESC) to cnacel"
8  DO WHILE (.NOT. PRINTSTATUS( ).AND. KEY<>27
9  KEY = INKEY ( )
10  ENDDO
11  DEACTIVATE WINDOW REP1
12  IF KEY = 27
13  RETURN TO MASTER
14  ENDIF
15  SET ESCAPE ON
16  RETURN
```

شرح إجراء التحقق من حالة الطابعة

١- السطر (٥) لإنشاء حقل ذاكرة متغير (Memvar) تخزن به القيمة العددية المقابلة للرمز الأمريكى القياسى (ASCII) للمفتاح الذى قام المستخدم بضغطه (سطر ٩).

٢- السطر (٧) لعرض رسالة داخل نافذة تظل معروضة على الشاشة مادامت الطابعة غير جاهزة للطباعة أو مادام المستخدم لم يضغط مفتاح (ESC) .

٣- السطر (٨) الى (١٠) حلقة دوارة ، ويظل الدوران مستمرا بين السطرين (٨) و (١٠) ، والعكس ، مادامت الطابعة غير جاهزة للطباعة أو لم يضغط المستخدم مفتاح (ESC) .

٤- من السطر (١٢) الى (١٤) حلقة شرطية تعنى أنه إذا قام المستخدم بضغط (ESC) فإنه يتم الغاء عملية الطباعة والإنتقال الى القائمة الرئيسية .

٥- فى حالة ضبط الطابعة وتجهيزها للطباعة وعدم ضغط (ESC) ، فإنه يتم الإنتقال الى السطر (١٥) الذى يعيد الوضع (ESC) الى الوضع (ON)

ويتم الإنتقال (بواسطة الأمر Return فى السطر ١٦) الى البرنامج الرئيسي للتقرير لمواصلة تنفيذ بقية إجراءات البرنامج الرئيسى .

نموذج إجراء إستقبال أوضاع الطباعة المناسبة للتقرير من المستخدم

```
1  PROCEDURE ENVIR
2  L MARGIN = 0
3  P LENGTH = 66
4  PSPACING = 1
5  PBPAGE = 1
6  PCOPIES = 1
7  PEPAGE = 9999
8  PEJECT = "NONE"
9  PWAIT = .F.
10  PQUALITY = .F.
11  PPITDCH = "DEFAULT"
12  DEFINE WINDOW ENVIR FROM 2,2 TO 23,78
13  ACTIVATE WINDOW ENVIR
14  @ 1,2 SAY "LEFT MARGIN:"GET LMARGIN;
15  PICTURE "99"
16  @ 3,2 SAY "PAGE LENGTH:"GET PLENGTH
17  @ 5,2 SAY"SPACE BETWEEN LINES:"GET PSPACING PICTURE"9"RANGE 1,3
18  @ 7,2 SAY "BEGIN PRINTING ON PAGE:"GET PBPAGE PICTURE "999"
19  @ 9,2 SAY "END PRINTING ON PAGE:"GET PEPAGE PICTURE "9999"
20  @ 11,2 SAY "NUMBER OF COPIES   :"GET PCOPIES PICTURE "999"
21  @ 13,2 SAY "EJECT PAPER"GET PEJECT PICTURE;
22   "@M BEFORE, AFTER, BOTH, NONE"MANAGE"PRESS SPACEBAR FOR
MORE CHOICE"
23 @ 15,2 SAY "WAITING BETWEEN PAGE"GET PWAIT PITCTURE "N"
24  @ 17,2 SAY "PITCH"GET PPITCH PICTURE "@M DEFAULT, PICA,;
25  ELITE, CONDENSED "MANAGE"PRESS SPACEBAR TO MORE CHOICES
26  @ 19,2 SAY "QUALITY PRINT"GET PQUALITY PICTURE "N"
27  READ
28  DEACTIVATE WINDOW ENVIR
29 _LMARGIN = LMARGIN
30 _PLENGTH = PLENGTH
31 _PSPACING = PSPACING
32 _PBPAGE = PBPAGE
33 _PEPAGE = PEPAGE
34  _PCOPIES = PCOPIES
35  _PEJECT = PEJECT
36  _PWAIT = PWAIT
37  _PPITCH =  PPITCH
38 _PQULAITY =  PQULAITY
39  RETURN
```

تعليق على الإجراء

١- من السطر (٢) الى (١١) لإنشاء حقول ذاكرة متغيرة لتخزين ما يدخله المستخدم من أوضاع مناسبة لاحتياجاته .
٢- من السطر (١٤) الى (٢٦) لاستقبال ما يدخله المستخدم من اختيارات وتخزينه فى حقول الذاكرة المتغيرة .
٣- من السطر (٢٩) الى (٣٨) لتعديل أوضاع حقول ذاكرة النظام بالأوضاع التى تم تخزينها داخل حقول الذاكرة المتغيرة .

الإجراء الخاص بوقف الطباعة عند الضغط على ESC

```
PROCEDURE RPEX
CONTNU = .F.
RETURN
```

يتم تعديل حقل الذاكرة المتغير المنطقى (Contnu) الى الوضع .F. فيتم الخروج من الحلقة الدوارة فى البرنامج الرئيسى .

نموذج إجراء التذييل (Footer)

```
PROCEDURE FOOTER
?
? "PAGE NUMBER:" AT 30
? -PAGNO
EJECT PAGE
RETURN
```

نموذج إجراء رأس الصفحة

```
PROCEDURE HEADER
??"DATE:" AT1
??":" AT 6
??"INVENTORY" AT 35
?
??"..............." AT 34
?
?
??"ITEM NO:" AT 0
??"ITEM NAME" AT 18
??" UNIT,," AT 58
??"BALANCE" A6 68
?
RETURN
```

نموذج إجراء تفاصيل التقرير

```
PROCEDURE DETAILS
?
?? NO AT 0
?
?? ITEMNAME PICTURE "XXXXXXX" AT 18
?
?? UNIT PICTURE "XXXXX" AT 58
?
?? BALANCE PICTURE "9999.99" AT 68
?
RETURN
```

تحتوى على أوضاع (Settings) خاصة بالإخراج والتحكم فى عملية الطباعة ، وتنشئ dBase IV هذه الأوضاع تلقائيا ، ويمكن الإحتفاظ بها فى ملف له الإمتداد (.prf) لاستخدامها مرة أخرى . وتسبق هذه المتغيرات علامة (_) لتمييزها عن بقية متغيرات الذاكرة .

ويمكن تقسيم هذه المتغيرات الى قسمين :

١- قسم يعمل أثناء عملية الطباعة مثل ...

_box, _pageno, _pcolno, _pform, _plength, _plineno, _pspacing, _tabs

٢- قسم لا يعمل أثناء عملية الطباعة ، ويشتمل على ثلاثة أقسام:

أ - قسم يتحكم فى أوضاع الطابعة مثل ...

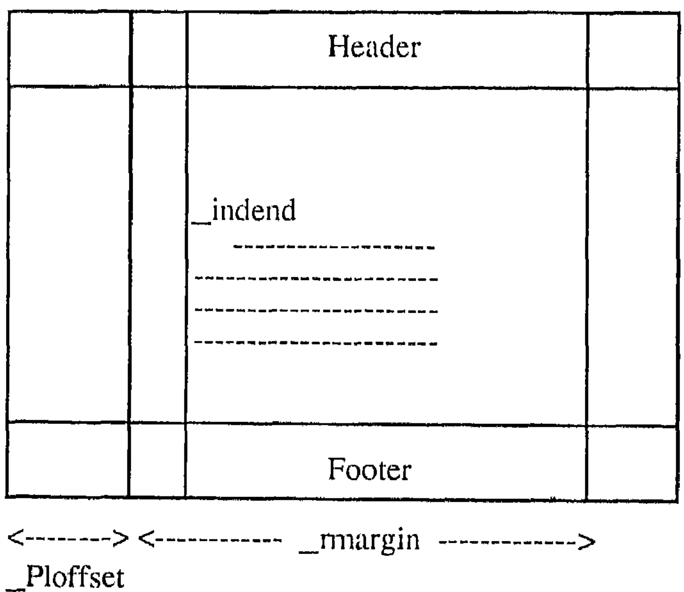
_padvance, _pdriver, _ploffset, _ppitch, _pquality, _pwait

ب - قسم المتغيرات التى تنفذ بواسطة الأمر PRINTJOB مثل ...

_pbpage, _pcopies, _pecode, _peject, _pepage, _pscode

جـ - قسم المتغيرات التى تؤثر على تشكيل النص مثل ...

_alignment, _indend, _lmargin, _rmargin, _wrap



* القسم الذى يشمل المتغيرات التى تعمل اثناء عملية الطباعة

١- _box

تتحكم فى إمكانية إظهار المربعات التى سبق أن عرّفتها بالأمر Define Box أو عدم إظهارها.

الصورة العامة:

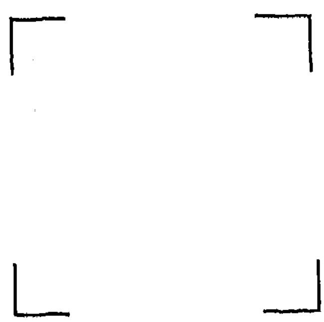
_box = <condition>

تستطيع من داخل البرنامج أن تحدد اللحظة التي تعرض فيها مربعا أو جزءا من مربع ، فمثلا تستطيع أن تكتب .box = .F_ قبل إتمام الطباعة فيتم طبع جزء فقط من المربع ، وتستطيع بعد ذلك فى لحظة ما أن تضع .box = .T_ ليتم طباعة جزء آخر من المربع .

مثال :

يمكن إظهار الأركان الأربعة فقط لمربع وذلك باستخدام الوظيف Space ( ) لحذف جزء من قاعدة المربع العليا والقاعدة السفلى للمربع ، وكذلك تعديل (_box = .F.) لعدم طباعة جزء من جانبى المربع

```
SET TALK OFF
USE PYRL
DEFINE BOX FROM 5 TO 75 HEIGHT 10 AT 5 DOUBLE
_BOX = .T.
POSITION = 1
SCAN
_BOX = (POSITION <4 .OR. POSITION >11)
?? NAME AT 7, NETSAL AT 50
?
POSITION = POSITION +1
ENDSCAN
?? SPACE (60) AT 8
```



٢- _pageno

تعطى رقم الصفحة الجارى طبعها <expN> = _pageno

الوضع المسبق (Default) : _pagno = 1

مثال :

يمكنك تحديد رقم صفحة بداية الطبع بأى رقم تبعا لاحتياجاتك ...

```
_pagno = 5
REPORT FORM REPORT 1 TO PRINTER
```

٣- _pcolno

يحرّك موضع الإخراج لعمود معين أثناء العرض على الشاشة أو أثناء الطباعة <expN> = _pcolno

أ - يمكنك تحديد _pcolno من صفر وحتى ٢٥٥

ب - يمكنك الرجوع الى نص سبق طباعته وإعادة الكتابة فوق جزء منه وذلك بوضع قيمة (pcolno-) أقل من القيمة الحالية بشرط وضع (_wrap = F.) وتكون الطابعة فى وضع ON .

جـ - الوظيفة ( ) PCOL تعطى القيمة الحالية لموقع رأس الطابعة .

مثال :

```
WRAP = .F.
?"This is just a test"
_PCOLNO = _PCOLNO - LEN("JUST A TEST")
?? REPLICATE ("=", LEN("JUST"))
?
```

تكون النتيجة ...

This is ~~JUST~~ a test

لاحظ أنه قد تمت الكتابة فوق الكلمة (JUST) .

٤- _plineno

يحدد رقم السطر الجارى طبعه <expN> = _plineno

الوضع المسبق : _plineno = 0

٥- _pspacing
تحدد المسافات التى تترك خالية بين السطور _pspacing = 1/2/3
الوضع المسبق : _pspacing = 1

٦- _tabs
تحدد وقفات (Stops) لجدول أو أكثر سواء للشاشة أو للطابعة ...
_tabs = <expC>

ملحوظة : tabs- تساوى أوضاع Tabs الموجودة فى ملف Config .db.

مثال :

_tabs = "2,10,30,60"

٧- _pform
يعطى إسم ملف (Print Form الحالى أو ينفذ ملف (Print Form يحتوى على أوضاع معينة للطباعة
_pform = "<print form filename>"

أ - الملف (.prf) أو (Print Form) هو ملف (Binary File) يحتوى على معلومات تخص أوضاع الطباعة .

ب - عند حفظ ملف (Report) أو ملف (Label) من مركز التحكم ، فإنه يمكن اختياريا حفظ أوضاع الطباعة للملف (.prf) وهذه الأوضاع يتم تحميلها تلقائيا فى أى وقت يتم طباعة أو تعديل ملف (Report) أو ملف (Label).

جـ - عند استخدام الأمر REPORT FORM أو LABEL FORM فإن الأوضاع المخزنة داخل الملف (.prf) لا تُحمَّل تلقائيا ، ولكن يجب أن تحدد أولا إسم الملف (.prf) للمتغير pform-

مثال :

_pform = "setting1"
report form report1

٨- _plength

تحدد طول الصفحة _plength = <expN>

الوضع المسبق : _plength = 66

* القسم الذى يشمل المتغيرات التى لا تعمل أثناء عملية الطباعة

◊ المتغيرات التى تتحكم فى أوضاع الطابعة

أ - padvance-

تحدد كيفية دفع الطابعة للورق _padvance ="Formfeed"/"linefeed"

الوضع المسبق : FORMFEED

FORMFEED يدفع الورق بمقدار النموذج

L.F. يدفع الورق بمقدار سطر

إذا أردت أن تطبع تقريرا يحتوى على صفحات قصيرة مثل شريط المرتبات مثلا لا يزيد طوله على ١٠ أسطر ، وبالتالى لا تستطيع أن تضبط هذا الوضع على الطابعة ولكن يمكنك عمل الآتى :

١- _plength = 10

٢- _padvance = "linefeed"

ب - _pdriver

تنفذ على مشغل وحدة (Driver) الطابعة المحددة ...

_pdriver = "<printer driver filename>"

جـ - _ploffset

تحدد بداية الهامش الأيسر للتقرير المطبوعة من حافة الصفحة اليسري ...

_ploffset = <expN>

تستطيع التحديد من صفر وحتى ٢٥٤

د - _ppitch

تحدد خطوة الطابعة [عدد الأحرف فى البوصة الواحدة - Char/inch] ، أو تعطى خطوة الطابعة المحددة حاليا .

_ppitch = "pica"/elite"/"condensed"/"default"

PICA ١٠ أحرف فى البوصة .

ELITE ١٢ حرفا فى البوصة .

CONDENSED ١٦ ، ١٧ حرفا/بوصة

DEFAULT تعني عدم تعديل للوضع المسبق .

هـ - _pqulaity

تختار بين مستويين للطباعة Draft, Quality ...

_pquality = <condition>

عندما تكون .F. يعنى الضبط على المستوى Draft .

و - _pwait

تحدد ما إذا كانت الطابعة تتوقف بعد طباعة كل صفحة أم لا ...

_pwait = <condition>

الوضع المسبق : .F.

◊ المتغيرات التى تنفذ بواسطة الأمر PRINTJOB

أ - -pbpage

تحدد صفحة البداية لتنفيذ الأمر Printjob ...

_pbpage = <expN>

الوضع المسبق : _pbpage = 1

تفيد فى حالة تعطل الطابعة بسبب انتهاء الورق أو لأى سبب آخر ، وتتيح لك أن تبدأ من حيث انتهت الطابعة قبل التوقف .

ب - _pcopies

تحدد عدد النسخ التى تطبع

_pcopies = <expN>

الوضع المسبق : _pcopies = 1

جـ - _pecode

_pecode = <expC>

تعطى شفرات التحكم لنهاية الطباعة ليقوم بتنفيذها الأمر Endprintjob . توضع -pecode قبل Endprintjob فى البرنامج ، ويقوم dBase IV بإرسال الشفرات المحددة بواسطة المتغير (_pecode) للطابعة بمجرد الوصول الى Endprintjob وتستخدم فى حالة الرغبة فى عمل تقرير بطرز (Styles) طباعة مختلفة حيث يقوم كل من المتغير (_pscode) و (_pecode) ببدء وإنهاء شفرات تحكم الطابعة التى تتيح طرزا مختلفة للطباعة .

د - _pscode

تعطى شفرات التحكم لبداية الطباعة ليقوم بتنفيذها الأمر Printjob ...

_pscode = <expC>

يوضع المتغير _pscode فى البرنامج قبل Printjob ، وتقوم dBase IV بإرسال شفرات التحكم المحددة بواسطة المتغير -pscode للطابعة عندما تقابل الامر Printjob . -pecode, -pscode تقوم بإرسال شفرات (codes) تحكم الطابعة التى تُبدئ أو تنهى طرز الطباعة (Typestyle) .

ملحوظة : ١- ??/? تستخدم فى البرنامج لتغيير طراز الطباعة فى كل سطر على حده .

٢- ??? تستخدم لتغيير طراز الطباعة فى فقرة .

٣ - _pecode, pscode تحددان طرز الطباعة العامة لما ينفذه الامر Printjob .

هـ - _pepage

تحدد صفحة النهاية للأمر Printjob ...

_pepage = <expN>

الوضع المسبق : ٣٢٧٦٧

المتغيرات التى تؤثر على تشكيل النص

١- _alignment

تتحكم فى الخرج الناتج من استخدام الأمرين ? و ?? إذا كان ضبط الأسطر

على اليسار أو ضبطها على اليمين أو التوسيط ...
_alignment = "LEFT"/"CENTER"/"RIGHT"

LEFT النص محاذ للهامش الايسر .
CENTER توسيط النص بين الهامشين الأيسر والأيمن .
RIGHT النص محاذ للهامش الايمن.

مثال :

_alignment = "RIGHT"
?? "DATE:", DATE ( )

يؤدى الى كتابة التاريخ على اليمين ...
DATE: 1-11-89

٢- _indent

تحدد الزحزحة للداخل لأول سطر من كل فقرة جديدة تطبع بواسطة الأمر (?) فى حالة أن يكون .-wrap = .T ...

_indent = <expN>

الوضع المسبق : _Indent = 0

٣- _imargin

تحدد الهامش الأيسر للخرج الناتج من استخدام الأمر (?) على أن يكون (_wrap = .T.)

_Imargin = <expN>

الوضع المسبق : _Imargin = 0

٤- _rmargin

تحدد الهامش الأيمن للخرج الناتج من استخدام أحد الأمرين ?? ,? ، على أن يكون (_wrap = .T.)

٥- _wrap

_wrap = <condition>

الوضع المسبق : .F.

عندما يكون الوضع .T. يتسبب فى أن يُطبع الخرج الناتج من استخدام الأمرين ?? ,? بين الهوامش المحددة بواسطة الأمرين _rmargin, _Lmargin

الفصل الثانى عشر

وظائف dBase IV

* وظائف خاصة بالزمن.
* وظائف خاصة بالتحويل.
* وظائف خاصة بالتمويل.
* وظائف رياضية.
* وظائف أخرى.

الفصل الثانى عشر

وظائف DBASE IV

أولا : وظائف خاصة بالزمن

١- CMONTH ( )

تعطى إسم الشهر من تعبير تأريخى معين ...

```
.? date ( )
01/07/89
. ? cmonth (date( ))
January
```

٢- DATE ( )

تعطى التاريخ كما هو مسجل فى نظام التشغيل (DOS)

```
. ? date ( )
01/07/89
.
```

٣- DAY ( )

تعطى رقم اليوم فى الشهر لتعبير تأريخى معين

```
.
. ? day (date ( ))
. 7
```

٤- DOW ( )

تعطى رقم اليوم فى الإسبوع فى تعبير تأريخى معين على اعتبار أن يوم الأحد (Sunday) هو رقم (١) فى الإسبوع .

```
.
. ? dow (date( ))
. 7
```

٥- CDOW ( )

تعطى إسم اليوم فى الإسبوع فى تعبير تأريخى معين

```
. ? cdow (date( ))
Saturday
.
.
```

٦- MONTH ( )

تعطى رقم الشهر فى تعبير تأريخى معين

```
.
. ? month (date ( ))
. 1
```

٧- TIME ( )

تعطى الوقت الحالى تبعا لما هو مسجل فى نظام التشغيل (DOS)

```
.
. ? time ( )
00:28:23
.
```

٨- YEAR ( )

تعطى العدد المقابل للسنة من تعبير تأريخى معين ، ويتكون من أربعة أرقام (أى أنه يحتوى على القرن أيضا)

```
.
. ? year ( )
. ? year (date ( ))
1989
.
```

ثانيا : وظائف خاصة بالتحويل

١- CHR ( )

تحوّل تعبيرا رقميا الى حرفيا .

```
. ? chr (65)
A
.
. ? chr (69)
E
.
. ? chr (62)
>
. ? chr (78)
N
```

٢- CTOD ( )

يحول تعبيرا حرفيا (Character) الى تعبير تاريخى (Date)

```
.
. mdate = "01/03/89"
01/03/89
. ? type ("mdate")
C
. mdate = ctod (mdate)
01/03/89
. ? type ("mdate")
D
.
```

٣- DMY ( )

يحول شكل التاريخ الى الشكل Day/Month/Year

```
. ? date ( )
07/01/89
. ? dmy (date ( ))
7 January 89
.
```

٤- DTOC ( )

تحول تعبيرا تأريخيا الى تعبير حرفى

```
.
.
. mdate = {11/11/89}
11/11/89
. ? type ("mdate")
D
. ms = dtoc (mdate)
11/11/89
. ? type ("ms")
C
.
```

٥- DTOR ( )

تحول الدرجات (Degrees) الى التدريج الدائرى (Radious)

```
. ? dtor (180)
3.14
.
```

٦- INT ( )

تحول أى رقم ذا كسر الى رقم صحيح

```
.
. ? int (344.76)
344
.
```

٧- FLOAT ( )

تحول أى عدد بنظام (BCD) الى عدد من النوع العلمى (float)

```
.
. ? float (.000675009)
.67500899E-3
.
```

٨- LOWER ( )

تحول الحروف الكبيرة الى حروف صغيرة

```
. ? lower ("MY NAME IS ALY")
my name is aly
```

٩- MDY ( )

تحول شكل التاريخ الى الشكل MonthDay, Year

```
. mdate = {01/08/89}
01/08/89
. ? mdy ("mdate")
. ? mdy (mdate)
August 01, 89
.
```

١٠- RTOD ( )

تحول التدريج الدائرى (Radious) الى درجات (Degree)

```
.
. ? dtor (180)
      3.141592653590
. ? rtod (3.141592653590)
180.00
.
```

١١- VAL ( )

تحول الأحرف الرقمية المخزنة فى حقل ذاكرة متغير الى أرقام

```
. store "25.42" to N
25.42
. ? type ("N")
C
. ? VAL (N)*2
    50.84
.
```

١٢- ABS ( )

تعطى القيمة المطلقة لعدد بغض النظر عن إشارته ، سالبة كانت أم موجبة .

```
.
. ? abs (-234)
234
.
```

ثالثا : وظائف خاصة بالعبارات

١- ASC ( )

تعطى رقما يمثل الرمز القياسى الأمريكى (ASCII) للحرف الأول من تعبير حرفى .

```
. ? asc ("NORA")
78
. ? asc ("ALY")
65
.
```

٢- CHR ( )

وهذه الوظيفة عكس الوظيفة السابقة ((ASC())

```
. ? chr (78)
N
.
```

٣- ISALPHA ( )

تستخدم لتحديد ما إذا كان تعبير معين يبدأ بحرف أم لا ، فإن هو كذلك تعطى (.T.)

```
.
. store "MAHA" to M
MAHA
. ? isalpha (M)
.T.
```

٤- ISLOWER ( )

تختبر ما إذا كان تعبير حرفى معين يبدأ بحرف صغير ، وفى تلك الحالة تعطى (.T.)

```
.
. ? islower (M)
.F.
.
```

٥- ( ) ISUPPER

تختبر ما إذا كان تعبير حرفى معين يبدأ بحرف كبير ، وفى تلك الحالة تعطى (.T.)

```
.
. ? isupper ("aly")
.F.
? isupper ("ALY")
.T.
```

٦- ( ) LEFT

تعطى جزءا من يسار تعبير حرفى معين أوحقل ذاكرة (Memo field) بعدد من الحروف يمثلها رقما تحدده

```
. ? left ("hassan",3)
has
```

٧- ( ) LEN

تعطى رقما يمثل عدد حروف تعبير حرفى معين أو لحقل ذاكرة

```
. ? len ("hassan")
6
```

٨- ( ) LTRIM

تزيل المسافات الخالية (Blanks) السابقة لتعبير حرفى معين

```
.
. mname1 = "    aly"
    aly
. mname2 = "hassan"
hassan
. ? ltrim (mname2)+mname1
hassan    aly
. ? ltrim (mname1)+mname2
alyhassan
.
```

٩- ( ) RIGHT

تعطى جزءا من تعبير حرفى معين يمثل رقما تحدده ويبدء من الحرف الأخير.

. ? Right ("HASSAN",3)
SAN

RTRIM ( ) -١٠

تزيل المسافات الخالية (Blanks) يمين تعبير حرفى معين

. mname = "aly   "
aly
. mname1 = "mahmoud"
mahmoud
. ? rtrim (mname)+mname1
alymahmoud

SUBSTR ( ) -١١

تعطي جزءا محددا من جملة أو عبارة يقوم المستخدم بتحديد موقع أول حرف من هذا الجزء وعدد حروفه

. ? substr ("how are you",5,3)
are

STUFF ( ) -١٢

تستبدل جزءا من عبارة بعبارة أخرى

stuff (<expC1>, <expN1>, <expN2>, <expC2>)

expC1 ... العبارة المراد استبدال جزء منها .

expC2 ... العبارة الجديدة التى ستحل محل جزء من العبارة الأصلية .

expN1 ... بداية الجزء من العبارة الذى سيتم استبداله .

expN2 ... عدد حروف العبارة الجديدة التى ستحل محل جزء من العبارة الأصلية .

.
. ? stuff ("my name is aly",12,4,"soha")
my name is soha

REPLICATE ( ) -١٣

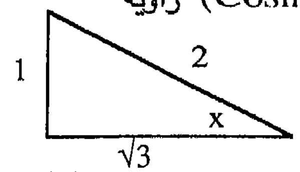
تكرر حرفا أو جملة عددا محددا من المرات

. ? replicate (chr (176),77)

رابعا : وظائف رياضيات

١- ACOS ( )

تعطى قيمة الزاوية بالتدريج الدائرى (Radians) لأى قيمة تمثل جيب تمام زاوية (Cosine)



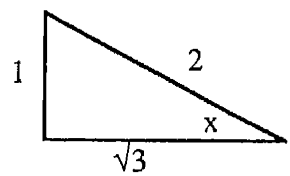
$$COS\ x = \frac{\sqrt{3}}{2}$$

لإيجاد قيمة الزاوية (x) مقيسة بالتدريج الدائرى

```
. store ((sqrt (3))/2) to n
. x = acos (n)
    0.523598775598
. x =acos (n)
0.523598775598
. ? rtod (x)
    30
```

٢- ASIN ( )

تعطى قيمة الزاوية بالتدريج الدرائرى لأى قيمة تمثل جيب الزاوية (Sine)



$$SIN\ x = \frac{1}{2}$$

لإيجاد الزاوية (x) مقيسة بالتدريج الدائرى

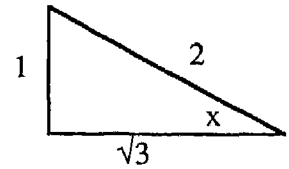
```
.
.
: sin x = .5
. x = asin (.5)
    0.523598775598
. ? rtod (x)
    30
```

٣- ATAN ( )

تعطى قيمة الزاوية مقيسة بالتدريج الدائرى لأى قيمة تمثل ظل (Tan) الزاوية (x)

$$\text{Tan } x = \frac{1}{\sqrt{3}}$$

$$x = \text{Atan } (\frac{1}{\sqrt{3}})$$



لإيجاد قيمة الزاوية (x) إذا علم قيمة ظل (Tan) الزواية

```
. Tan x = 1/(3^.5)
. store 1/(3^.5) to n
       0.577350269190
. x = atan (n)
       0.523598775598
. ? rtod (x)
      30
```

ATN2 ( ) -٤

تعطى قيمة الزاوية مقيسة بالتدريج الدائرى (Radians) إذا علم قيمة جيب الزاوية (Sine) ، وجيب تمام الزاوية (Cosine) .

مثال:

لو علمنا أن قيمة جا (Sine) x مقيسة بالتدريج الدائرى = ٠,٥ ، وأن قيمة جتا (Cos) x مقيسة بالتدريج الدائرى = ٠,٨٦ ، فإن قيمة الزاوية x مقيسة بالتدريج الدائرى =

```
. store dtor (.5) to n
     0.008726646260
store dtor (.86) to m
     0.015009831567
. ? atn2 (n,m)
x = 0.526627271434 [in radians]
. ? rtod (.5266272714334)
x = 30.173520029624[in degrees]
```

COS ( ) -٥

تعطى قيمة جيب تمام الزاوية (Cosine) لأى زاوية مقيسة بالتدريج الدائرى .

```
. store dtor (60) to n
     1.047197551197
. store cos (n) to m
     0.500000000000
```

٦- LOG ( )

تعطى قيمة اللوغاريتم (Log) لرقم محدد .

```
.
.
? log (3.22)
     1.169381359556
.
. ? log (10)
     2.302585092994
```

٧- LOG 10 ( )

تعطى قيمة اللوغاريتم لرقم للأساس (Base) ١٠.

```
. ? log 10 (3.23)
     0.509202522331
. ? log 10 (1)
     0
```

٨- MIN ( )

تعطى القيمة الأصغر بين رقمين أو تاريخين أو تعبيرين .

```
.
. ? min (7778,654)
     654
.
```

٩- MAX ( )

تعطى القيمة الأكبر بين رقمين أو تاريخين أو تعبيرين .

```
. ? MAX (6574,3432)
     6574
```

MOD ( ) -١٠

تعطى القيمة المتبقية من ناتج قسمة عددين ٠

```
.
. ? mod (7,3)
1
```

RTOD ( ) -١١

يحوّل التدريج الدائرى الى درجات (Degrees) .

```
. ? rtod (.5266272714334)
30.173520029624 degrees
```

SIGN ( ) -١٢

تعطى قيمة تمثل إشارة (Sign) عدد معين ؛

١ ، للعدد الموجب

-١ ، للعدد السالب.

(٠) ، للصفر

```
x = -37
? sign (x)
-1

. x = -25
-25
. ? sign (x)
-1
. z = 99
99
. ? sign (z)
1
. m= 0
. ? sign (m)
0
```

١٣- SIN ( )

تعطى قيمة جيب زاوية (Sin) مقيسة بالتدريج الدائرى (Radians) .

```
. ? sin (3.14)
    0.001592652916
```

١٤- SQRT ( )

تعطى الجذر التربيعى لأى عدد موجب .

```
.
. ? SQRT (64)
    8
```

١٥- TAN ( )

تعطى قيمة ظل الزاوية (Tan) لزواية معينة مقيسة بالتدريج الدائرى .

```
.
. ? tan (3.14)
    -0.001592654936
```

١٦- ROUND ( )

تقرّب الأرقام العشرية لعدد محدد من الكسور العشرية .

```
.
. ? round (865.67555,3)
    865.676
```

لتقريب الرقم الى أقرب ٣ كسور عشرية.

خامسا : وظائف أخرى

١- ALIAS ( )

تعطى الإسم الرمزى لمنطقة ذاكرة (Memory work area)

٢- BAR ( )

القائمة الرأسية تتكون من مجموعة من الإختيارات ، والوظيفة BAR ( ) تعطى رقم آخر اختيار تم انتقاؤه من القائمة .

٣- BOF ( )

تشير لبداية ملف قاعدة البيانات

```
MNO = 0
DO WHILE .T.
@ 10,6 SAY "1=ITEM BEFORE 2 = ITEM AFTER 3 = ANY ITEM 4 =;
EXIT" GET MNO
DO CASE
CASE MNO = 1
SKIP -1
IF BOF ( )
GO TOP
ENDIF
DO SCREEN
CASE MNO = 2
SKIP
IF EOF ( )
GO BOTTOM
ENDIF
DO SCREEN
CASE MNO = 3
SEEK MNO
IF .NOT. FOUND ( )
? "THIS ITEM NOT FOUND!"
WAIT ""
LOOP
ENDIF
ENDCASE
ENDDO
```

٤- COL ( )

تعطى رقم العمود لموقع المؤشر الحالى ...

مثال :

```
? col ( )
25
```

٥- DBF ( )

تعطى إسم ملف قاعدة البيانات المفتوح فى منطقة الذاكرة المفتوحة .

```
: USE STORE
? DBF ( )
Store
```

٦- DISKSPACE ( )

تعطى مقدار المساحة غير المشغولة (المتاحة) على القرص الممغنط .

٧- EOF ( )

تشير لنهاية ملف قاعدة البيانات .

٨- FIELD ( )

تعطى إسم حقل إذا عرف رقمه فى بنية ملف قاعدة البيانات .

```
. USE STORE
? FIELD (8)
. UNIT
```

٩- FILE ( )

تتحقق من وجود ملف معين على الدليل الحالى ، وتعطى الإجابة المنطقية (.T.) فى حالة وجود الملف ، والإجابة (.F.) فى حالة عدم وجوده .

مثال :

```
? file ("C:store .dbf")
.T.
```

يجب أن يكون اسم الملف مقرونا بمشغل الوحدة (Drive) ، والإمتداد (Extension) الخاص بالملف .

IS COLOR ( ) - ١٠

تعطى فى حالة أن تكون الشاشة المستخدمة ملونة .

? Is color ( )
.T.

MEMORY ( ) -١١

تعطى مقدار المساحة غير المشغولة فى ذاكرة الحاسب الرئيسية (RAM) .

? memory ( )
220124

MENU ( ) -١٢

تعطى اسم القائمة الحالية .

? menue ( )
mbar

OS ( ) -١٣

تعطى إسم نظام التشغيل الحالى .

? os ( )

PAD ( ) -١٤

تعطى اسم آخر اختيار تم التعامل معه من القائمج الأفقية الحالية .

? pad ( )
Data

POPUP ( ) -١٥

تعطى إسم القائمة الرأسية الحالية .

? popup ( )
DPOP

١٦- RECCOUNT ( )

يستنتج عدد السجلات فى ملف قاعدة البيانات .

```
. use store
? reccount ( )
30
```

١٧- RECNO ( )

تعطى رقم السجل الحالى فى ملف قاعدة البيانات المفتوح .

```
. use store
. ? recno ( )
10
```

١٨- RECSIZE ( )

تعطى المقدار المشغول من القرص الممغنط لسجل فى ملف قاعدة بيانات مفتوح .

```
. use store
. ? recsize ( )
80
```

الفصل الثالث عشر

* خطوات إنشاء البرنامج .
* تصميم قواعد البيانات .
* ملاحظات حول إنشاء البرنامج الرئيسى .
* ملاحظات حول إنشاء برنامج الاضافة .
* ملاحظات حول إنشاء البرامج الفرعية والاجراءات .
* ملاحظات حول إنشاء برامج الاستفسار .
* ملاحظات حول إنشاء برامج التقارير .
* تنفيذ برامج كتبت بلغات أخرى من خلال برنامج dBase.
* حماية البيانات والبرامج .
* طرق تنفيذ البرنامج .
* كيف تزيد من سرعة أداء البرنامج ؟.

الفصل الثالث عشر

كيف تنشئ برنامجا باستخدام dBASE IV

تتميز dBase IV بوجود العديد من الأدوات (Tools) التى تيسر على المستخدم وتنوب عنه فى كتابة الأوامر لإنجاز مهام معينة ومن أمثلة هذه الأدوات منتج التطبيقات ، مصمم الإستفسارات ، مصمم نماذج الأشكال ، ومصمم التقارير ، ومصمم العناوين . وعند استخدام أحد هذه الأدوات ، فإنه يكفى أن يقوم المستخدم بوصف ما يريد لتقوم الأداة نيابة عنه بكتابة الأوامر اللازمة لإنجاز المهمة .

ولكن ، الى أى حد يمكن استخدام هذه الأدوات ؟ وهل يمكن أن تغنى هذه الأدوات عن المبرمج ذى المهارات المتقدمة ؟
والجواب بالطبع ... لا ، فعلى الرغم من أن هذه الأدوات تيسر حتى على المبرمج الخبير ، إلا أنها لا تغنى عن مهارات البرمجة التى لا يمكن إكتسابها إلا بمواصلة الدراسة التخصصية ، بالإضافة الى الخبرة المكتسبة نتيجة الممارسة الطويلة فى إنشاء البرامج .

وهذا الفصل محاولة لسد الثغرة بين استخدام هذه الأدوات ، وإنجاز برامج تتميز بمهارات خاصة .

خطوات إنشاء البرنامج

١- تحديد المهام التى سوف ينجزها البرنامج وكذلك تحديد شكل المخرجات .
٢- تحليل المهام ، وبناء عليه يتم الآتى :
أ - تحديد تصور مبدئى للإجراءات أو البرامج الفرعية التى تقابل هذه المهام ، ويكون هذا التصور عبارة عن لغة وسيطة بين لغة البرمجة والجمل العادية .
ب - تصميم قواعد البيانات المناسبة لهذه المهام .
٣- أ - إنشاء الإجراءات أو البرامج الفرعية اللازمة بحيث يقوم كل إجراء أو برنامج فرعى بتنفيذ أحد مهام البرنامج .
ب - إختبار كل إجراء أو برنامج فرعى على حدة للتأكد من صلاحيته وخلوه من الأخطاء .
٤- تجميع كل هذه البرامج الفرعية أو الإجراءات ، والتحكم فيها من خلال القائمة الرئيسية للبرنامج التى يتم إنشاؤها داخل البرنامج الرئيسى .
٥- إختبار البرنامج على عينات من البيانات والتأكد من خلوه من الأخطاء .

تصميم قواعد البيانات

عند تصميم قواعد البيانات يراعى دائما عدم استخدام الملفات الكبيرة نظرا لأن البحث والوصول الى البيانات فى هذه الملفات يكون بطيئا ، لذلك يجب أن تجزأ الملفات الكبيرة الى عدد من الملفات الصغيرة ، ويراعى كذلك عدم تكرار أى حقول بين هذه الملفات ، وتحديد الرابطة الصحيحة بين الملفات وهى الرابطة التى على أساسها تتم فهرسة الملفات .

ولفهم ذلك اليك المثال التالى :

نفترض أن لدينا ملف أحد المكتبات (ملف أ مثلا) يحتوى على الحقول التالية :
رقم الكتاب - إسم الكتاب - نوع الكتاب - إسم المؤلف - جنسية المؤلف - رقم عضوية المستعير - إسم المستعير - المهنة - العنوان - الجنسية ...
نلاحظ من هذه الحقول أنه يمكن تقسيم مهامها الى قسمين ، قسم يختص بمعلومات عن الكتاب ، وقسم يختص بمعلومات عن المستعير ، ولذلك يمكن تجزئة الملف (أ) الى ملفين (ب) و (ج) ، ويتم ربط الملفين باستخدام الحقل

المفتاح (رقم الكتاب)

الملف (ب)

رقم الكتاب	إسم الكتاب	نوع الكتاب	إسم المؤلف	جنسية المؤلف

الرابطة

الملف (ج)

رقم الكتاب	رقم العضوية	إسم المستعير	المهنة	العنوان	الجنسية

نلاحظ من الملف (ب) أنه يمكن تقسيمه الى قسمين : قسم خاص بمعلومات عن الكتاب ، وقسم خاص بمعلومات عن المؤلف ، ولذلك يمكن تجزئة الملف (ب) الى ملفين (ب١) و (ب٢) ، وتكون الرابطة بين الملفين على الحقل (رقم الكتاب).

الملف (ب١)

رقم الكتاب	إسم الكتاب	نوع الكتاب

الملف (ب٢)

رقم الكتاب	إسم المؤلف	جنسية المؤلف

كذلك يمكن تجزئة الملف (ج) الى ملفين (ج١) و (ج٢) ، وتكون الرابطة على أساس الحقل (رقم العضوية) هذه المرة ، نظرا لأن الرابطة بين الملفين المذكورين تكون على أساس تقديم بيانات عن المستعير.

الملف (ج١)

رقم العضوية	رقم الكتاب	إسم المستعير

الملف (ج٢)

رقم العضوية	العنوان	الجنسية

ملاحظات حول إنشاء البرنامج الرئيسى

١ - البرنامج الرئيسى هو البرنامج الذى يتحكم فى كل الجزئيات البرمجية فى البرنامج سواء كانت إجراءات أو برامج فرعية ، وهو البرنامج الذى يتم فيه تهيئة وسط البرنامج وإنشاء القوائم . ومن المفضل أن يتم إنشاء البرنامج الرئيسى كاملا من خلال منتج التطبيقات وذلك بغرض السرعة حيث أن ذلك لا يستغرق منك أكثر من ساعة على أكثر تقدير .

وهذا الإسلوب مناسب للمبتدئين ،المتقدمين على السواء ، ولكن يلاحظ أن البرنامج المنشأ بواسطة منتج التطبيقات يحتوى على كثير من الإجراءات التى قد لا تلائم كل البرامج ، وعلى الرغم من أنها لا تنفذ أثناء تنفيذ البرنامج ، إلا أنها تُحمل فى ذاكرة الحاسب مما يتسبب عنه إشغال حيّز من الذاكرة ، بالإضافة الى أن ذلك يبطئ من سرعة تنفيذ البرنامج ، ولذلك فيستطيع المتقدمون أن يحذفوا هذه الإجراءات التى يرون أنها زائدة عن حاجة البرنامج .

ويجب الحذر من حذف أحد الإجراءات الأصلية أو أحد حقول الذاكرة المتغيرة ، ولذلك فمن الأفضل وضع علامة (*) قبل بداية كل سطر يراد حذفه ثم يختبر البرنامج جيدا ، وعند حدوث أى خطأ فإنك تستطيع إعادة النظر مرة أخرى فى الأجزاء المراد حذفها . ويجب أن لا يحاول المبتدئون التعديل فى برنامج منتج التطبيقات .

٢ - أحيانا وأثناء تنفيذ البرنامج المنشأ باستخدام منتج التطبيقات ، قد تعرض هذه الرسالة :

Outrange Window

ومعناها أنه أثناء تنفيذ أحد الإجراءات أو البرامج الفرعية التى تم إنشاؤها خارج منتج التطبيقات قد تم تجاوز حدود النافذة (Savescr) المنشأة بواسطة منتج التطبيقات .

وللافاة ذلك ، عدّل الإحداثى الأفقى للإطار (Savescr) من الإحداثى (٢١) الى الإحداثى (٢٣) أو الإحداثى (٢٤) فى برنامج منتج التطبيقات .

ملاحظات حول إنشاء برنامج الإضافة

فى المعتاد يتم استقبال البيانات وتخزينها داخل حقول ذاكرة متغيرة أو مصفوفة ، وعند التأكد من صحة البيانات المدخلة يتم إضافتها لنهاية ملف قاعدة البيانات .

وعند إنشاء المصفوفة التى تقوم بهذه المهمة يتم تعريف عناصر المصفوفة بناء على حقول قاعدة البيانات ، ولفهم ذلك اليك المثال التالى :

لإضافة بيانات الى ملف (Student .dbf) الذى يحتوى على الحقول التالية :

1- Number
2- Name
3- Add
4- Tel

فإنه يتم إعلان المصفوفة كما يلى :

Declare M[4]

أى مصفوفة أحادية لها ٤ عناصر تقابل عدد ٤ حقول لملف قاعدة البيانات (Student) ، ثم يتم تعريف كل عنصر من عناصر المصفوفة كما يلى :

M[1] = 0

أى يكون العنصر الأول من المصفوفة حقل ذاكرة متغير رقمى يقابل حقل قاعدة البيانات Number .

M[2] = space (20)

العنصر الثانى فى المصفوفة يماثل حقل (NAME) فى قاعدة البيانات أى يكون من نفس النوع (Character) وله نفس الطول .

M[3] = space (15)

العنصر الثالث حرفى ليماثل حقل Add فى ملف قاعدة البيانات .

M[4] = 0

العنصر الرابع رقمى ليماثل حقل Tel في ملف قاعدة البيانات .

ثم يتم استقبال البيانات من المستخدم وتُخزن داخل عناصر المصفوفة [M] كما يلى :

@ 5,5 SAY "STUDENT NUMBER" GET M[1] PICTURE "999";

لاحظ أنه تم استقبال بيانات المستخدم عن رقم الطالب Number وخزنت داخل العنصر M[1] .

ثم بعد التأكد من صحة إدخال هذه البيانات ، ورغبة المستخدم فى الإحتفاظ بها داخل الملف يستخدم الأمر التالى :

APPEND FROM ARRAY M

ليتم إضافة بيانات المصفوفة الى سجل جديد يضاف لنهاية ملف قاعدة البيانات المفتوح .

نقاط هامة :

١- أحيانا ، وعند استخدام نماذج أشكال (Format) كبيرة لإدخال أو لتعديل البيانات قد تظهر الرسالة التالية :

Insufficient memory

وتعنى أنك قد استنفدت المواقع المخصصة داخل الذاكرة ، وللافاة ذلك ، غير الوضع المسبق فى ملف Config .db :

من Bucket = 2

الى Bucket = 3 أو 4

ويمكنك الزيادة حتى ٣١.

٢- يجب أن تستخدم ضوابط التأكد من البيانات المدخلة لحظيا فى نفس وقت إدخال البيانات .

ويمكنك ذلك باستخدام Valid أو Range أو picture أو Function ، وكذلك استخدام وظائف المستخدم (UDF) مع Valid للتحقق من الشروط المركبة لصحة بيانات الحقول المختلفة .

٣- يجب أن تعمل على أن تقلل فرصة إمكانية إدخال المستخدم لبيانات خطأ كلما استطعت الى ذلك سبيلا كما يلى :

أ - حاول أن تجعله يختار أحد الإختيارات المعروضة أمامه ، وذلك باستخدام كشوف القيم (Values lists) [إنظر جزء منتج التطبيقات] .

ب - بعمل رمز رقمي (كود) للبيانات [وذلك إن أمكن حصرها] ، فمثلا تستطيع أن ترمز الإدارات المختلفة فى أحد الهيئات أو الشركات ، بحيث تعطى لكل إدارة رمزا رقميا محددا مع إمكانية عرض شاشة بهذا الرمز متى ما أراد المستخدم ، وذلك بضغط F1 مثلا.

مثال :

```
ON KEY LABEL F1 DO APPENDX
MDEP = 0
@ 10,5 SAY "ENTER DEPART.:"GET MDEP
READ
PROCEDURE APPENDX
X = INKEY ( )
ACTIVATE WINDOW APP
DO CASE
CASE VARREAD ( ) = "MDEP"
?
TEXT
00 = FINANCE DEPART
01 = COMMERC DEPART
.
.
ENDTEXT
.
.
CASE VARREAD ( ) = "MJOB"
.
.
ENDCASE
? INKEY (0)
DEACTIVATE WINDOW APP
RETURN
```

جـ - إن أمكن حصر المدخلات فى عدد قليل من الخيارات ، يمكنك كتابة :

```
MUNIT = SPACE (3)
@ 5,5 SAY "ENTER UNIT:"GET MUNIT;
PICTURE "@M KG, KM, CM, TON, M";
MESSAGE "PRESS SPACEBAR TO MORE CHOICES"
READ
```

وعند التنفيذ سيعرض أمام حقل MUNIT الإختيار الأول وهو (Kg) ، وعند ضغط مسطرة المسافات يستبدل بالاختيار الثانى وهو (Km) ، وهكذا ...
وعند ضغط <Enter> يتم اختيار الخيار المعروض .

ملاحظات حول إنشاء البرامج الفرعية والإجراءات

يتكون البرنامج فى dBase من برنامج رئيسى ، وعدد من البرامج الفرعية أو الإجراءات . وتتميز الإجراءات عن البرامج الفرعية بالميزات التالية :

١- توضع الإجراءات فى ذاكرة الحاسب لذلك يكون الأداء أسرع فى التنفيذ .

٢- تتميز الإجراءات بالمرونة فى الإستخدام عن البرامج الفرعية حيث يمكن استخدام مجموعة المعاملات (Parameters) مع الإجراءات ، وبذلك يمكن استخدام الإجراء الواحد لتنفيذ أحد المهام بأكثر من كيفية ، وبذلك يكون استخدام الإجراءات استخداما ديناميكيا مرنا فى حين تفتقر البرامج الفرعية لذلك .

٣- التحفظ الوحيد فى استخدام الإجراءات هو سعة ذاكرة الحاسب إذ يجب أن تضع فى الإعتبار مناسبة عدد الإجراءات المستخدمة مع سعة ذاكرة الحاسب ، وإلا ظهرت الرسالة التالية :

. Insufficient Memory

ولكن مادامت سعة الذاكرة تكفى فإنه يمكنك أن تستخدم ملف إجراءات يحتوى على حتى ١١٧٠ إجراءً للملف الواحد كحد أقصى .

ملاحظات حول إنشاء برنامج الإستفسار

تقدم dBase IV وسيلة جيدة لتصميم الإستفسارات باستخدام شاشة تصميم الإستفسارات التى تستخدم أسلوب الإستفسار باستخدام الأمثلة (Query by examples - QBE) ، ولكنك قد ترغب فى كتابة أوامر dBase IV فى برنامجك مباشرة دون الرجوع الى ملف الإستفسار المنشأ بواسطة شاشة تصميم الإستفسارات .

ويستطيع المتقدمون ذلك بكل سهولة ، أما المبتدئون فبإمكانهم أيضا كتابة برامج الإستفسارات بكفاءة تامة كما يلى :

١- إنشئ ملف الإستفسار بالمواصفات التى تريدها باستخدام شاشة تصميم الإستفسارات كما سبق شرحها فى فصل الإستفسارات فى الجزء الأول ، ولنفرض أن الملف الذى أنشأته هو باسم Test .

٢- من نقطة الأوامر اكتب الآتى :

```
COPY FILE TEST .qbe TO TEST .prg
```

٣- فيصبح لديك الآن ملف لبرنامج يحتوى على الأوامر التى تنشئ الإستفسار كما سبق أن حددتها ، وتستطيع فتح هذا الملف ورؤية محتوياته كما يلى :

MODIFY COMMAND TEST

وتستطيع أيضا تنفيذه باستخدام الأمر DO :

. DO TEST

ملاحظات حول إنشاء برنامج التقرير

تقدم dBase IV أيضا وسيلة على مستوى عال من الكفاءة والمرونة لإنشاء كافة أنواع التقارير التى تناسب جميع الأغراض باستخدام شاشة تصميم التقارير (Report design screen) ولذلك ينصح بعدم تضييع الوقت والجهد فى كتابة برامج التقارير ، ولكن يمكنك بكفاءة تامة وفى وقت وجيز جدا أن تنشئ تقريرا كاملا باستخدام شاشة تصميم التقارير . ولرؤية الأوامر التى تم إنشاؤها نتيجة استخدام مصمم التقارير لإنشاء تقرير اسمه TEST1 ... مثلاً من نقطة الأوامر اكتب ...

MODIFY COMMAND TEST1. FRG

ويمكنك التعديل فى هذا الملف ولكن ضع فى الاعتبار بأن أمر LOCATE يتم تنفيذه على البرنامج على الرغم من عدم كتابته داخل الملف .

ملحوظة : أحياناً قد تظهر هذه الرسالة : " Printer not ready " عند حدوث أى خطأ فى التوافق بين الطابعة و dBase IV ، ويمكنك التغلب على ذلك بكتابة الأمر التالى :

SET PRINTER TO FILE PRN

تنفيذ برامج كتبت بلغات أخرى من خلال برنامج dBase IV

يمكنك تنفيذ برامج كتبت بلغات أخرى من خلال نقطة الأوامر أو من خلال البرنامج كما يلى :

١- تنفيذ برنامج كتب بلغة Assembly ...

أ - حمّل البرنامج فى الذاكرة باستخدام الأمر Load

LOAD <binary filename>

ب - لتنفيذ البرنامج يكتب الأمر

CALL <filename>[with<experssion list>]

٢- تنفيذ برنامج كتب بأى لغة أخرى

تستطيع مثلا أن تنفذ برنامج كتب بلغة Basic من خلال dBase IV باستخدام الأمر Run أو ! .

ولتنفيذ برنامج (Prog1) الذى كتب بلغة Basic ، يتم إنشاء ملف (Batch file) يحتوى على السطر : BASIC PROG1
وليكن اسم هذا الملف أيضا (Prog1) ، ولتنفيذ هذا البرنامج من خلال برنامج Dbase يكتب الآتى :

. RUN PROG1

كيف تنشئ شاشة تحتوى على معلومات إيضاحية مساعدة HELP ؟

يمكنك إنشاء شاشة معلومات إيضاحية لمساعدة المستخدم فى استخدام البرنامج باستخدام الأمر : ON KEY LABEL F1 DO <filename>

وبمجرد ضغط المستخدم على مفتاح F1 يتم تنفيذ البرنامج (Filename) الذى يعرض شاشة تحتوى على معلومات إيضاحية لمساعدة المستخدم .
إنظر كيفية إنشاء هذا البرنامج فى الفصل (١٤) .

كيف تحمى بياناتك وبرامجك ؟

تقدم dBase IV وسيلة فعالة فى حماية ملفاتك باستخدام الأمر PRO-TECT ، ولكن يجب أن لا يستخدم هذه الإمكانية إلا المتقدمون . وفيما يلى بعض الملاحظات التى يمكن أن تستخدمها فى حماية بياناتك وبرامجك ...

١- تحتوى Dbase IV على برنامج مترجم (Compiler) يحوّل ملف البرنامج الذى تم كتابته ويسمى الملف المصدر (Source file) الى ملف مكتوب بلغة الآلة يسمى (Object file) . ولكى تضمن أن لا يطلع أحد على برامجك تستطيع أن تحذف ملف البرنامج المصدر Source كما يلى :
نفترض أن لدينا ملف برنامج يسمى Test ، فعند البحث فى الدليل (Dir) ستجد ثلاثة ملفات لهم نفس الإسم (Test) وكل منهم له الإمتداد (Extension) الخاص به كالآتى :
Test .prg ملف البرنامج المصدر .
Test .BAK ملف البرنامج الإحتياطى .
Test .dbo ملف البرنامج المترجم (Object).

أ – أحذف الملف Test .prg

Delete file test .prg

ب – أحذف الملف Test .bak

Delete file test .bak

وبذلك يكون لديه فقط ملف البرنامج المترجم الذى له الإمتداد (dbo) مكتوبا بلغة الآلة حيث لا يستطيع أحد أن يطلع على محتوياته .

ملحوظة : يجب أن تحتفظ بنسخة من ملف البرنامج المصدر (source file) لديك حتى تستطيع التعديل فيه لأنك لا تستطيع أن تعدل فى الملف المترجم.

٢– إنشاء كلمة السر (Password) للبرنامج بحيث لا تسمح لأى شخص (لا يعرف كلمة السر) باستخدام البرنامج أو جزء معين منه مثل برنامج الحذف مثلا كالتالى :

```
SET ESCAPE OFF
PW = SPACE (5)
DO WHILE .NOT, PW<>"ALY00"
PW = SPACE (5)
@ 5,5 say "please enter the password:"GET PW;
color w*/N,N/N
READ
ENDDO
```

مادام المستخدم لم يكتب كلمة السر وهى هنا ALY00 فلن يتم الإنتقال للجزء التالى لهذا البرنامج .

كيف تنفذ البرنامج ؟

تستطيع تنفيد البرنامج مباشرة من نظام التشغيل (DOS) كما يلى :

أولا: تشغيل البرنامج بمجرد تشغيل الحاسب ، وذلك بإنشاء ملف (Autoexec .bat) الذى ينفذ بمجرد تشغيل الحاسب .

ويتم إنشاء ملف Autoexec .bat كما يلى :

```
C>copy con:   Autoexec .bat
ECHO OFF
Date
Time
CD\dbase
dbase test
CD\
```

إضغط F6 ثم اضغط <enter>
وبمجرد تشغيل الحاسب يتم تنفيذ البرنامج Test بدون أى تدخل من المستخدم .

ثانيا: تشغيل البرنامج (Test) من نظام التشغيل (DOS) ...
أ - من الدليل الرئيسى (Root Directory)
وذلك بإنشاء ملف (Batch .file) كما يلى :

```
C>copy con:  test .bak
echo off
CD\dbase
dbase test
echo on
```

إضغط F6 ثم <enter>
وينفذ البرنامج بكتابة Test بعد تحميل نظام التشغيل (DOS) من الدليل الرئيسى (Root Dir) .
ب - من الدليل الموجود به dBase IV
وذلك بإنشاء ملف (Batch file) كما يلى :

```
C>copy con: test. bat
echo off
dbase test
echo on
```

إضغط F6 ثم <enter>
وينفذ البرنامج (Test) بكتابة Test من الدليل الموجود به نظام dBase IV

كيف تزيد من سرعة أداء برنامجك ؟

كما علمنا يوجد فى dBase IV برنامج مترجم compiler يقوم بتحويل البرنامج الذى تم كتابته باستخدام أوامر dBase ويسمى البرنامج المصدر source file إلى برنامج مكتوب بلغة الآلة يسمى object file حيث يكون أسرع فى التنفيذ .

ولزيادة سرعة تنفيذ البرنامج توجد فى dBase IV إمكانية لربط اجراءات البرنامج المختلفة فى برنامج واحد .

ولتنفيذ ذلك من نقطة الأوامر أكتب :

```
. COMPILE < filename >
```

ومن دليل dBase أكتب ما يلى :

```
C> DBLINK < filename> / L / D
```

تستخدم D/ لإظهار رسالة على الشاشة فى نهاية كل عملية ربط .

وسيتم إنشاء ملف (تلقائياً) له نفس إسم ملف البرنامج وله الإمتداد (TXT.) يحتوى على معلومات عن جميع البرامج الفرعية والإجراءات المستخدمة .

الفصل الرابع عشر

أمثلة عمليـــــــــة

الفصل الرابع عشر

أمثلة عمليـــــــة

فيما يلى أمثلة عملية مع شرح كامل لها ، من الممكن أن تتخذها نموذجاً عند إنشاء برامج خاصة بك .

١ - المثال الأول :

فى المثال التالى نموذجاً للبرنامج الرئيسى بالإضافة إلى بعض البرامج الفرعية أو الإجراءات الشائعة فى معظم البرامج مثل برنامج الإضافة أو التعديل أو الحذف .

البرنامـــج الرئيســى

يتكون البرنامج فى dBase IV كما علمنا من برنامج رئيسى (Main program) ، ومجموعة من الإجراءات (Procedures) ، أو البرامج الفرعية ، وكما سبق القول ، فإن البرنامج الرئيسى يتم فيه الآتى :

١ - تجهيز الوسط المناسب (Environment) .
٢ - إنشاء حقول الذاكرة المتغيرة (إن وجدت) .
٣ - إنشاء القوائم .
٤ - التحكم فى كل الإجراءات أو البرامج الفرعية .
٥ - إنهاء البرنامج وإغلاق جميع الملفات وإعادة تهيئة الوسط .

```
**************************************************
* program ......: ALY.PRG
* Author ........: ENGINEER: ALY MAHMOUD BAHLOUL
* Date ...........: 07/1/89
**************************************************
CLEAR ALL
CLEAR WINDOWS
CLOSE ALL
CLOSE PROCEDURE
SET BELL ON
SET CARRY OFF
SET CENTURY OFF
SET CLOCK OFF
SET CONFIRM OFF
SET DEVICE TO SCREEN
SET ESCAPE OFF
SET ECHO OFF
SET MESSAGE TO ""
SET PRINT OFF
SET SAFETY ON
SET TALK OFF
SET INSTRUCT OFF
SET COLOR TO
CLEAR
SET SCOREBOARD OFF
SET STATUS OFF
DO MENUDEF
SET PROCE TO YBAR
DO YBAR
SET CONFIRM OFF
SET DELIMITER OFF
ON KEY LABEL F1
CLEAR ALL
CLEAR WINDOWS
CLOSE ALL
CLOSE PROCEDURE
 SET CLOCK OFF
SET ESCAPE ON
SET MESSAGE TO ""
CLEAR
RETURN
```

من هذا البرنامج الرئيسى (Aly .PRG) يُلاحظ الآتى :

* الأسطر من (٦) الى (٢٦) لتجهيز الوسط المناسب للبرنامج .

١- السطر (٦) لمسح الذاكرة الرئيسية (RAM) من كل حقول الذاكرة المتغيرة (فيما عدا حقول ذاكرة النظام طبعا) وكل القوائم ، وكذلك غلق جميع الملفات .

٢- السطر (٧) لإزالة جميع النوافذ من الشاشة ومن الذاكرة .

٣- السطر (٨) لإغلاق جميع الملفات .

٤- السطر (٩) لغلق ملفات الإجراءات (Procedure files) .

٥- السطر (١٠) لجعل جرس الحاسب مهيئا للعمل .

٦- السطر (١١) لعدم نسخ بيانات آخر سجل للسجل التالى فى حالة استخدام الأمر Append أو الأمر Insert .

٧- السطر (١٢) لعدم وضع القرن ضمن التاريخ .

٨- السطر (١٣) لعدم إظهار الساعة على الشاشة .

٩- السطر (١٤) لانتقال المؤشر (Curser) تلقائيا من حقل الى حقل تالى بمجرد امتلاء الحقل الأول .

١٠- السطر (١٥) لضمان عدم إرسال بيانات الى الطابعة كنتيجة لاستمرار الأمر (Set device to print) من برنامج سابق .

١١- السطر (١٦) لضمان عدم الخروج من البرنامج إذا تم الضغط على مفتاح (ESC) أثناء تنفيذ البرنامج .

١٢- السطر (١٧) لعدم إظهار سطر الأمر أثناء تنفيذه .

١٣- السطر (١٨) لحذف أى رسالة (Message) على الشاشة .

١٤- السطر (١٩) لوقف طبع أى بيانات حتى يتم تغيير الوضع فى البرنامج باستخدام (Set print on) .

١٥- السطر (٢٠) لإعطاء رسالة تحذيرية فى حالة نسخ ملف على ملف موجود فعلا أو فى حالة إنشاء ملف جديد له نفس إسم ملف قديم .

١٦- السطر (٢١) لعدم إظهار نتيجة تنفيذ بعض الأوامر مثل إنشاء حقول الذاكرة المتغيرة (Memvars) أو الأمر SUM أو COPY .

١٧- السطر (٢٢) لعدم إظهار الرسائل (Prompts) الخاصة بعمليات الشاشة الكاملة .

١٨- السطر (٢٣) لضبط الألوان تبعا لسابق ضبطها فى ملف Config .db .

١٩- السطر (٢٤) لمسح الشاشة .

٢٠- السطرين (٢٥) و (٢٦) لعدم إظهار التعبيرات التى تصف بعض الحالات الراهنة مثل Del أو Caps أو Ins .

٢١- السطر (٢٧) لتنفيذ إجراء تعريف قوائم البرنامج .

٢٢- السطر (٢٨) لفتح ملف الإجراءات (YBAR) .

٢٣- السطر (٢٩) لتنفيذ الإجراء المسؤول عن تنفيذ القوائم .

٢٤- من السطر (٣١) الى السطر (٤٠) خاصة بانهاء البرنامج وإعادة ضبط الوضع المسبق .

٢٥- السطر (٤١) للإنتقال الى المكان الذى نفذ منه البرنامج سواء كان نقطة الأوامر أو من لوحة التطبيقات بمركز التحكم .

```
*********************************
PROCEDURE MENUDEF
IF ISCOLOR ( )
SET COLOR OF NORMAL TO W+/B
SET COLOR OF MESSAGES TO W+/N
SET COLOR OF TITLES TO W/B
SET COLOR OF HIGHLIGHT TO GR+/BG
SET COLOR OF BOX TO GR+/BG
SET COLOR OF INFORMATION TO B/W
SET COLOR OF FIELDS TO N/BG
ENDIF
CLEAR
DEFINE WINDOW ALYHLP FROM 0,0 TO 21,79
@ 24,00 SAY " "
@ 23,00  SAY "PLEASE..."
SET BORDER TO DOUBLE
DEFINE MENU YBAR MESSAGE 'Move with: '+CHR (27)+CHR (26)+' -;
press <Enter> to select choice - <F1> Help'
DEFINE PAD DATA OF YBAR PROMPT "DATA    " AT 1,4
ON PAD DATA OF YBAR ACTIVATE POPUP YDATA
DEFINE PAD TRANSACT OF YBAR PROMPT "TRANSACTION"  AT 1,19
ON PAD TRANSACT OF YBAR ACTIVATE POPUP YTRAN
DEFINE PAD SEARCH OF YBAR PROMPT "SEARCH" AT 1,39
ON PAD SEARCH OF YBAR ACTIVATE POPUP YSEARCH
DEFINE PAD MANT OF YBAR PROMPT "MAINT
ON PAD MAINT OF YBAR ACTIVATE POPUP YMAINT
DEFINE PAD EXIT OF YBAR PROMPT "EXIT" AT 1,71
ON SELECTION PAD EXIT OF YBAR DO ALYPROG1
?? "W"
SET BORDER TO DOUBLE
DEFINE POPUP YDATA FROM 4,1 TO 10,21;
 MESSAGE 'MOVE WITH: '+CHR (27)+CHR (26)+CHR (24)+';
 PRESS:  '+CHR (17)+CHR (196)+CHR (217)+' TO CHOOSE ..;
HELP: F1'
DEFINE BAR 1 OF YDATA PROMPT "APPEND ITEM"
DEFINE BAR 2 OF YDATA PROMPT "--------------------"  SKIP
DEFINE BAR 3 OF YDATA PROMPT "MODIFY ITEM"
DEFINE BAR 4 OF YDATA PROMPT "--------------------"  SKIP
DEFINE BAR 5 OF YDATA PROMPT "DELETE ITEM"
ON SELECTION POPUP YDATA DO ALYPROG2
?? "A"
SET BORDER TO DOUBLE
DEFINE POPUP YTRAN FROM 4,19 TO 12,44 ;
 MESSAGE 'MOVE WITH: '+CHR (27)+CHR (26)+CHR (25)+CHR (24)+';
```

```
PRESS: '+CHR (17)+CHR (196)+CHR (217)+'  TO CHOOSE.. HELP: F1'
DEFINE BAR 1 OF YTRAN PROMPT "START POINT BALANCE"
DEFINE BAR 2 OF YTRAN PROMPT "------------------------------" SKIP
DEFINE BAR 3 OF YTRAN PROMPT "TRANSACT BALANCE"
DEFINE BAR 4 OF YTRAN PROMPT "------------------------------" SKIP
DEFINE BAR 5 OF YTRAN PROMPT "ITEM BALANCE NOW"
DEFINE BAR 6 OF YTRAN PROMPT "----------------------------" SKIP
DEFINE BAR 7 OF YTRAN PROMPT "ITEM BALANCE AT ANY TIME"
ON SELECTION POPUP YTRAN DO ALYPROG3
?? "I"
SET BORDER TO DOUBLE
*-- POPUP
DEFINE POPUP YSEARH FROM 4,37 TO 8,58 ;
MESSAGE 'MOVE WITH: '+CHR (27)+CHR (26)+CHR (25)+CHR (24)+';
PRESS: '+CHR (17)+CHR (196)+CHR (217)+' TO CHOOSE .. HELP: F1 '
DEFINE BAR 1 OF YSEARH PROMPT "SEARCH BY ITEMNAME"
DEFINE BAR 2 OF YSEARH PROMPT "----------------------" SKIP
DEFINE BAR 3 OF YSEARH PROMPT "SEARCH BY ITEMNAME"
ON SELECTION POPUP YSEARH DO ALYPROG4
?? "T"
SET BORDER TO DOUBLE
DEFINE POPUP YMAINT FROM 4,46 TO 12,79 ;
MESSAGE 'MOOVE WITH: '+CHR (27)+CHR (26)+CHR (25)+CHR (24)+';
PRESS: '+CHR (17)+CHR (196)+CHR (217)+' TO CHOOSE.. HELP: F1'
DEFINE BAR 1 OF YMAINT PROMPT "COPY BALANCES TO FLOPPY DISK"
DEFINE BAR 2 OF YMAINT PROMPT "---------------------"  SKIP
DEFINE BAR 3 OF YMAINT PROMPT "RESTORE BALANCE FROM FLOPPY
 DISK"
DEFINE BAR 4 OF YMAINT PROMPT "---------------------"  SKIP
DEFINE BAR 5 OF YMAINT PROMPT "REPORTS          "
DEFINE BAR 6 OF YMAINT PROMPT "---------------------"  SKIP
DEFINE BAR 7 OF YMAINT PROMPT "RE- INDEX FILE"
ON SELECTION POPUP YMAINT DO ALYPROG5
@ 23,00 CLEAR
RETURN
```

من الإجراء (Menudef) يلاحظ الآتى :

١- لاحظ أن الإجراء يبدأ بالأمر (Procedure Menudef) فى سطر (٤٤) وينتهى بالأمر (Return) فى سطر (١٢٠) .
٢- من السطر (٤٥) وحتى (٥٣) لضبط ألوان القوائم وما يتبعه من إجراءات فى حالة أن تكون الشاشة ملونة (IF ISCOLOR( )) وإن لم تكن ملونة فإنه سيتم تطبيق الألوان المسبق ضبطها فى ملف Config .db .
٣- السطر (٥٤) لمسح الشاشة .
٤- السطر (٥٥) لتعريف النافذة (Window) الخاصة بتقديم معلومات مساعدة (Help) إيضاحية عن البرنامج .
٥- السطر (٥٦) لمسح أى جملة أو رسالة موجودة على السطر الأخير للشاشة.
٦- الأسطر (٥٧) و (٧١) و (٨٢) و (٩٥) و (١٠٥) و (١١٨) لهم مهمة واحدة هى أنه أثناء تنفيذ إجراء تعريف القوائم ، فإنه يتم تحميل ذاكرة الحاسب (RAM) بتلك القوائم لتكون قابلة للتنفيذ فى أى لحظة باستخدام الأمر (Activate) ، وكنتيجة لهذا التحميل تجد أن الحاسب يتأخر لبضع ثوان لتحميل تعريف كل قائمة فى الذاكرة ، ولإعطاء رسالة للمستخدم تطمئنه وتذكره بأنه يتم تحميل البرنامج ، فقد وضعت الكلمة (Please) فى السطر (٥٧) ليتم عرضها على الشاشة فى السطر (٢٣) ثم بعد الإنتهاء من تعريف القائمة الأولى يضيف الحرف (W) ، وبعد الإنتهاء من تعريف القائمة الثانية يضيف (A) وبعد الإنتهاء من الثالثة (I) ثم الحرف (T) بذلك تتم كتابة الكلمة (Wait) ببطء ، وبعد الإنتهاء من كتابتها على الشاشة يكون الحاسب قد انتهى من تحميل تعريفات القوائم بالذاكرة .
٧- السطر (٥٨) لتجعل أحرف المستطيلات والقوائم الرأسية مزدوجة .
٨- السطر (٥٩) لتعريف القائمة الأفقية (YBAR) ولاحظ Chr (27) و Chr (26) لتعطى مقابل هذين الرقمين طبقا لنظام الـ (ASCII) ، وينتج منهما رسم السهم لليمين والسهم لليسار .
٩- السطر (٦١) لتعريف الإختيار (Data) فى القائمة الأفقية (YBAR) .
١٠- لتعريف الفعل المترتب عن اختيار (Data) فى القائمة الأفقية وهو تنفيذ القائمة الرأسية (YDATA) .

١١- من السطر (٦٣) الى (٧٠) يتم تعريف بقية اختيارات القائمة الأفقية (YBAR) والأفعال المترتبة على هذه الإختيارات .

١٢- السطر (٧٣) لتعريف القائمةالرأسية (YDATA) ، لاحظ أن Chr (24) وChr (25)خاصة برسم السهم لأعلى ولأسفل ، وChr (17) و Chr (196) و Chr (217) خاصة برسم علامة (Enter) .

١٣- السطر (٧٦) خاص بتعريف الإختيار الأول من القائمة الرأسية وهو الإختيار (Append Item) .

١٤- السطر (٧٧) خاص برسم خط فاصل بين اختيارين فى القائمة الرأسية ، والأمر (Skip) يعنى أن لا يقف المؤشر عند هذا الخط .

١٥- السطر (٨١) يعنى أنه عند انتقاء أى اختيار من القائمة الرأسية فإنه يتم تنفيذ الأجراء Alyprog2 .

١٦- حتى السطر (١١٧) تكرار لنفس الخطوات السابقة لتعريف بقية القوائم الرأسية .

```
*******************************************
* Program......:  YBAR.PRG ****PROCEDURE FILE**
*******************************************
PROCEDURE YBAR
DO ADJUST
ACTIVATE MENU YBAR
@ 0,0 CLEAR TO 2,79
RETURN
*******************************************
PROCEDURE ADJUST
ON KEY LABEL F1 DO ALYHLP1
    @ 0,0 TO 2,79 DOUBLE COLOR GR+/BG
    @ 1,5 SAY "DATA  " COLOR W+/N
    @ 1,20 SAY "TRANSACTION" COLOR W+/N
    @ 1,40 SAY "SEARCH " COLOR W+/N
    @ 1,55 SAY "MAINT  " COLOR W+/N
    @ 1,72 SAY "EXIT" COLOR W+/N
RETURN
*******************************************
PROCEDURE ALYPROG1
DO CASE
CASE "DATA" = PAD ( )
    DO YDATA
CASE "TRANSACT" = PAD ( )
    DO YTRAN
CASE "SEARCH" = PAD ( )
    DO YSEARH
CASE "MAINT" = PAD ( )
    DO YMAINT
CASE "EXIT" = PAD ( )
    CLOSE DATABASES
    QUIT
ENDCASE
SET MESSAGE TO
RETURN
```

الإجراء (YBAR) خاص بتنفيذ القائمة الأفقية (YBAR)

١- السطر (٥) فى الإجراء (YBAR) لتنفيذ الإجراء (ADJUST) الخاص بتحسين شكل القائمة الأفقية وذلك بعمل إطار (السطر ١٢) حول اختيارات القائمة الأفقية . والسطور من (١٣) الى (١٧) لتحسين شكل القائمة الأفقية أثناء الإنتقال بين الإجراءات المختلفة ، وكذلك أثناء إنهاء البرنامج .

٢- الإجراء (ALYPROG1) خاص بتنفيذ الأفعال أو الإجراءات المترتبة على انتقاء إحدى خيارات القائمة الأفقية .

٣- السطر (١١) لعرض شاشة (Help) مساعدة لشرح أجزاء البرنامج عند قيام المستخدم فى أى وقت أثناء تنفيذ البرنامج بضغط F1 .

```
36 ************************************************
37 PROCEDURE YDATA
38 ON KEY LABEL F1 DO ALYHLP1
39 @ 22,0
40 ACTIVATE POPUP YDATA
41 RETURN
42 ************************************************
43 PROCEDURE ALYPROG2
44 DO CASE
45 CASE BAR ( ) = 1
46      DO RPA
47 CASE BAR ( ) = 3
48      DO ED
49 CASE BAR ( ) = 5
50      DO DEL
51 ENDCASE
52 SET MESSAGE TO
53 RETURN
```

الإجراء (YDATA) خاص بتنفيذ القائمة الرأسية (YDATA) .

١- السطر (٣٨) خاص بعرض شاشة المساعدة (Help) الخاصة بالقائمة الرأسية (YDATA) .

٢- السطر (٤٠) لتنفيذ القائمة الرأسية (YDATA) .

٣- الإجراء (Alyprog2) خاص بتنفيذ الإجراءات المترتبة على اختيار أحد اختيارات القائمة الرأسية (YDATA) .

لاحظ أن الإجراء (Alyprog2) يتم استدعاؤه وتنفيذه من السطر (٨١) فى الإجراء (Menudef) .

```
54 ************************************
55 PROCEDURE YTRAN
56 ON KEY LABEL F1 DO ALYHELP1
57 ACTIVATE POPUP YTRAN
58 RETURN
59 ************************************
60 PROCEDURE ALYPROG3
61 DO CASE
64 CASE BAR ( ) = 3
65   DO PRG2
66 CASE BAR ( ) = 5
67   DO PRG4
68 CASE BAR ( ) = 7
69   DO PRG5
70 ENDCASE
71 SET MESSAGE TO
72 RETURN
```

١- الإجراء (YTRAN) خاص بتنفيذ القائمة الرأسية (YTRAN) .

٢- الإجراء (Alyprog3) خاص بتنفيذ الإجراءات الناتجة عن انتقاء أحدى خيارات القائمة الرأسية (YTRAN) .

ويتم استدعاء هذا الإجراء وتنفيذه من السطر (٩٤) فى الإجراء (Menudef) .

```
73 ************************************
74 PROCEDURE YSEARH
75 ON KEY LABEL F1  DO ALYHLP1
76 ACTIVATE POPUP YSEARH
77 RETURN
78 ************************************
79 PROCEDURE ALYPROG4
80 DO CASE
81 CASE BAR ( ) = 1
82      DO SE1
83 CASE BAR ( ) = 3
84      DO SE2
85 ENDCASE
86 SET MESSAGE TO
87 RETURN
```

١- الإجراء (YSEARCH) خاص بتنفيذ القائمة الرأسية (YSEARCH) .

٢- الإجراء (Alyprog4) خاص بتنفيذ الإجراءات الناتج عن اختيارات القائمة الرأسية (YSEARCH) .

يتم استدعاء وتنفيذ الإجراء (Alyprog4) من السطر (١٠٤) من الإجراء (Menudef) .

```
***************************************
PROCEDURE YMAINT
ON KEY LABEL F1 DO ALYHLP1
ACTIVATE POPUP YMAINT
RETURN
***************************************
PROCEDURE ALYPROG5
DO CASE
CASE BAR ( ) = 1
 DO MA1
CASE BAR ( ) = 3
 DO MA2
CASE BAR ( ) = 5
 DO MA3
CASE BAR ( ) = 7
 DO MA4
ENDCASE
SET MESSAGE TO
RETURN
```

١- الإجراء (YMAINT) خاص بتنفيذ القائمة الرأسية (YMAINT) .

٢- الإجراء (Alyprog5) خاص بتنفيذ الإجراء الناتج عن انتقاء أحد خيارات القائمة الرأسية (YMAINT) .

```
121 *-----------------------------------------------
122 PROCEDURE ALYHLP1
123 ACTIVATE WINDOW ALYHLP
124 ACTIVATE SCREEN
125 @ 0,0 CLEAR TO 21,79
126
127 @ 24,0 SAY space (79)
128 @ 24,26  SAY "Please Press any key to continue ..." COLOR W+*/N
129 @ 0,0 SAY ""
130 x = INKEY ( )
131   @ 2,2 SAY "THIS SCREEN TO EDIT HELP TEXT."
132   @ 4,2 SAY "ENG. ALY MAHMOUD BAHLOUL"
133   x = INKEY (0)
134 DEACTIVATE WINDOW ALYHLP
135 RETURN
136 *-----------------------------------------------
```

١- الإجراء (Alyhlp1) خاص بعرض شاشة معلومات إيضاحية مساعدة لإجزاء البرنامج فى أى وقت عند ضغط F1 أثناء تنفيذ البرنامج ، ويمكن عمل إجراء لعرض معلومات مساعدة لكل قائمة على حدة .

٢- السطر (١٢٣) خاص بتنفيذ النافذة (Alyhlp) على الشاشة والسابق تعريفها فى سطر (٥٥) فى الإجراء (Menudef) .

٣- السطر (١٢٤) للخروج من تأثير النافذة (Alyhlp) مع عدم إزالتها من الشاشة .

٤- السطر (١٢٥) لمسح مستطيل من الشاشة ركنه الأيسر العلوى عند (صفر و صفر) ، وركنه الأيمن السفلى عند (٢١ ، ٧٩) .

٥- السطر (١٢٧) لمسح السطر (٢٤) من الشاشة .

٦- السطر (١٢٩) لإعادة وضع المؤشر (Cursor) فى أعلى يسار الشاشة .

٧- السطر (١٣٠) لمسح الوسيط الخازن (Buffer) .

٨- السطر (١٣١) و (١٣٢) لعرض نص بيانات الإيضاح المساعدة (Help) .

٩- السطر (١٣٣) لوقف (Pause) الشاشة حتى يقوم المستخدم بضغط أى مفتاح للخروج من شاشة الإيضاح (Help) الى نفس المكان الذى تم الضغط عنده على F1 .

١٠- السطر (١٣٤) لإزالة النافذة (Alyhlp) من الشاشة .

عند تنفيذ البرنامج السابق يتم عرض القائمة المبينة فى شكل (١٣ – ١) أدناه.

DATA	TRANSACTION	SEARCH	MAINT	EXIT

APPEND ITEM
MODIFY ITEM
DELETE ITEM

MOVE WITH ↓↑ PRESS ↵ TO CHOOSE .. HELP : F1

شكل (١٣ – ١)

نموذج برنامج الإضافة

هذا البرنامج الفرعى (RPA) يتم تنفيذه عند انتقاء (ADD ITEM) من القائمة الرأسية (YDATA) المتصلة بالإختيار (DATA) من القائمة الأفقية (YBAR) .

```
CLEAR
USE STORE ORDER NO
SET PROCEDURE TO TYPE
DEFINE WIND A FROM 5,10 TO 10,68
DO WHILE .T.
MNO = 0
MCL = 0
MDL = 0
MUNIT = 0
MNAME = SPACE (50)
MTYP = 0
DO ASCREEN
READ
IF READKEY ( ) < 256
CLEA
RETURN
ENDIF
IF MNO > 0
CH = " "
ACTIV WIND A
?? CHR (7)
@ 0,1 SAY " IS THIS RIGHT " W+*/N
@ 2,0 SAY " DO YOU WANT SAVE THIS DATA ? (Y/N)?" GET CH! ;
VALID CH$ " YyNn " ERROR " PLEASE ENTER YES OR NO ONLY " ;
 COLOR W+/N
READ
DEAC WIND A
IF CH$ " Yy "
REBLACE NO WITH MNO, NAME WITH MNAME, DL WITH ;
 MDL, CL WITH MCL
ENDIF
ENDIF
ACTIV WIND A
?? CHR (7)
CH = "  "
@ 1,5 SAY "DO YOU WANT ADD NEW ITEMS ? (Y/N)" GET CH
 VALID CH$"YyNn";
ERROR "PLEASE ENTER Y OR N ONLY"
```

```
READ
DEAC WIND A
IF CH$ = " Yy "
APPEND  BLANK
LOOP
ELSE
CLEAR
EXIT
SET PROCEDURE TO
ENDIF
IF EOF ( )
GO BOTTOM
ENDIF
ENDDO
SET PROCEDURE TO
RETURN
*----------------
PROCEDURE ASCREEN
@ 1,1 TO 23,78 DOUBLE
@ 2,30 TO 4,50
@ 3,32 SAY " ITEM DATA "
@ 6,4 SAY  " ITEM No.  :" GET MNO PICT "999999" VALID DUPLICATE (MNO);
MESS " PLEASE ENTER THE ITEM NUMBER "
@ 8,4 SAY " ITEM NAME  :" GET MNAME WHEN MNO > 0;
MESS " PLEASE ENTER THE ITEMNAME "
@ 10,4 SAY " UNIT  :" GET MUNIT RANG 1,5 WHEN MNO > 0;
MESS " 1=M .. 2=KM .. 3=TON .. 4=No .. 5=KG "ERROR " PLEASE ENTER A NUMBER FROM 1 TO 5 "
@ 12,4 SAY " TYPE  :" GET MTYP RANG 1,17 WHEN MNO > 0;
MESS " PLEASE ENTER A NUMBER FROM 1 - 17 "
@ 14,4 SAY " REORDER LVL  :" GET MCL WHEN MNO > 0;
RETURN

FUNCTION DUPLICATE
PARAMETERS NUM
DEC = .F.
IF NUM = 0
RETURN .NOT. DEC
ENDIF
SEEK NUM
IF FOUND ( )
ACTIVATE WINDOW A
?? CHR (7)
? " THIS NUMBER ALREADY EXIST "
```

```
? " PLEASE RE- ENTER "
WAIT  " PRESS ANY KEY "
DEACTIVATE WIND A
RETURN DEC
ELSE
RETURN .NOT. DEC
ENDIF
```

من البرنامج (RPA) نلاحظ الآتى :

١- السطر (١) لمسح الشاشة .
٢- السطر (٢) لفتح الملف (Store) وجعل الفهرس المتحكم هو (No) .
٣- فتح ملف الإجراءات (Type) .
٤- السطر (٤) لتعريف النافذة (A) .
٥- بدء حلقة دوارة لا نهائية تبدأ من السطر (٥) وتنتهى فى السطر (٦١) .
٦- من السطر (٦) الى (١١) لإنشاء حقول الذاكرة المتغيرة (Memvar) .
٧- السطر (١٢) لاستدعاء وتنفيذ الإجراء (Ascreen) الذى يبدأ من السطر (٦٥) ، وهو خاص بتصميم شاشة إدخال البيانات . ومن هذا الإجراء (Ascreen) نلاحظ الآتى :
أ - فى السطر (٦٩) يتم التحقق من عدم إزدواجية رقم الصنف باستخدام الوظيفة التى يعرّفها المستخدم DUPLICATE (UDF)
ب - السطر (٧١) لا يمكن إدخال بيانات لهذا الحقل مادام رقم الصنف فى الحقل الأول يساوى أو أصغر من صفر .
٨- السطر (١٣) لتهيئة شاشة إدخال البيانات لقبول بيانات من المستخدم .
٩- من السطر (١٤) الى (١٧) حلقة شرطية تعنى أنه إذا لم يتم إدخال أى بيانات فى شاشة إدخال البيانات (READKEY ( ) < 256) سوف يتم الإنتقال الى البرنامج المستدعى ، وفى هذه الحالة يتم الإنتقال الى القائمة الأفقية الرئيسية فى البرنامج .
١٠- السطر (٢٠) لتنفيذ النافذة (A) لعرض الرسالة فى سطرين (٢٢) و (٢٣).
١١- سطر (٢٦) لإزالة النافذة (A) من الشاشة .
١٢- من سطر (٢٧) الى (٢٩) حلقة شرطية تعنى أنه فى حالة رغبة المستخدم فى الإحتفاظ بالبيانات التى تم إدخالها فإنه يتم حفظها داخل ملف قاعدة البيانات (Store) باستخدام الأمر (Replace) .
١٣- من السطر (٣١) الى (٣٧) تنفيذ النافذة (A) التى تعرض رسالة وينتظر رد المستخدم عليها .
١٤- من السطر (٣٨) الى (٤٥) حلقة شرطية تعنى أنه فى حالة رغبة المستخدم فى إضافة صنف آخر ، فإنه يتم إضافة سجل جديد (سطر ٣٩) ، ثم الإنتقال لبداية الحلقة الدوارة اللا نهائية فى سطر (٦) لإعادة عرض شاشة إدخال البيانات بالنسبة للسجل الجديد .

وفى حالة عدم رغبة المستخدم فى إضافة صنف آخر (سطر ٤١) ، فإنه يتم مسح الشاشة (سطر ٥٣) والخروج من الحلقة الدوارة (سطر ٤٣ الى ٥٠) .

١٥- السطر (٤٤) لغلق ملفات الإجراءات .

١٦- من السطر (٤٦) الى (٤٨) حلقة شرطية تعنى أنه فى حالة وجود مؤشر السجلات (Pointer) فى آخر الملف (EOF) فإنه يتم الإنتقال الى السجل الأخير فى الملف.

١٧- السطر (٥٠) لغلق ملفات الإجراءات .

١٨- السطر (٥١) للإنتقال الى القائمة الأفقية (YBAR) .

شرح وظيفة المستخدم (DUPLICATE)

هذه الوظيفة تعطى رسالة تحذيرية عند قيام المستخدم بإدخال رقم الصنف بنفس رقم صنف آخر سبق إدخاله ، وتمنع قبول ما يدخله المستخدم باستخدام(Valid) .

١- السطر (٧١) لعرض المعطيات المستخدمة مع الوظيفة (Duplicate) والمعطيات فى هذا المثال تكون هى رقم الصنف (Mno) .

٢- السطر (٧٢) لإنشاء حقل ذاكرة متغير لتخزين الحرف المنطقى (.F.) .

٣- من السطر (٧٣) الى (٧٥) حلقة شرطية تعنى أنه فى حالة إدخال المستخدم رقم الصنف = صفرا ، فسوف يتم إعطاء نتيجة الوظيفة إيجابية (.T.) ، ويتم الإنتقال الى الحقل التالى فى شاشة إدخال البيانات .

٤- السطر (٧٦) يبحث فى الملف المفهرس عن السجل الذى يحتوى على رقم الصنف مساوٍ لما أدخله المستخدم .

٥- من السطر (٧٧) الى (٩٦) حلقة شرطية تعنى أنه فى حالة العثور على سجل فى الملف يحتوى على نفس رقم الصنف فسوف يتم عرض رسالة داخل نافذة مع إحداث صوت الجرس تفيد بأن هذا الرقم موجود فعلا ، وتعطى الوظيفة نتيجة سلبية (.F.) تمنع قبول هذا الرقم .

وفى حالة عدم العثور على سجل يحتوى على نفس رقم الصنف ، فسوف تعطى الوظيفة نتيجة إيجابية (.T.) ويتم قبول ما أدخله المستخدم داخل حقل الذاكرة المتغير (Mno) .

نموذج برنامج الحذف (Del)

يتم استدعاء وتنفيذ برنامج الحذف عند انتقاء المستخدم للإختيار (Delete item) من القائمة الرأسية (YDATA) المتصلة بالإختيار (Data) فى القائمة الأفقية (YBAR) .

```
CLEAR
USE STORE ORDER NO
DO WHILE .T.
DEFINE WIND A FROM 10,5 TO 17,60
ACTIVA WIND A
MNO = 9
@ 1,0 SAY "------ DELETE ----"
@ 3,5 SAY " ENTER THE ITEME NUMBER(0 TO EXIT) " GET MNO
READ
DEAC WIND A
IF MNO = 0
EXIT
ENDIF
ACTI WIND A
DES = "  "
@ 5,5 SAY " ARE YOU SURE ( Y/N ) GET DESVALID DES$ " YyNn " ERROR ;
 " PRESS  (Y)  OR ( N)  ONLY "
READ
DEAC WIND A
SEEK MNO
IF .NOT. FOUND ( )
?? CHR (7)
ACTIV WIND  A
@ 4,5 SAY" THIS NUMBER NOT EXIST .. PLEASE RE-ENTER "
WAIT " PRESS ANY KEY "
DEACT WIND A
LOOP
ENDIF
DELETE
IF DES$ ' Yy '
PACK
ELSE
RECAL
ENDIF
ACTIV WIND A
DES1 = "  "
```

```
36  @ 3,5 SAY " DO YOU WANT DELETE ANOTHER ... " GET DESVALID;
    DES$ " YyNn " ERROR ;  " PRESS (Y) OR ( N) ONLY "
37  READ
38  DEACTI WIND A
39  IF DES1$ "Yy "
40  LOOP
41  ELSE
42  EXIT
43  ENDIF
44  ENDDO
45  RETURN
```

من البرنامج (DEL) يُلاحظ التالى :

١- السطر (١) لمسح الشاشة .

٢- السطر (٢) لفتح الملف (Store) مع جعل الفهرس المتحكم هو الفهرس (No) .

٣- السطرين (٣) و (٤٤) لإنشاء الحلقة الدوارة اللا نهائية .

٤- السطر (٤) لتعريف النافذة (A) .

٥- السطر (٥) لتنفيذ النافذة (A) على الشاشة مع عرض رسالة داخل النافذة (سطر ٧ و ٨) ، واستقبال اختيار المستخدم داخل حقل الذاكرة المتغير الذى سبق إنشاؤه فى سطر (٦) .

٦- من السطر (١١) الى (١٣) حلقة شرطية تعنى أنه إذا أدخل المستخدم فى حقل رقم الصنف (صفرا) ، فإنه يتم الخروج من الحلقة الدوارة الرئيسية باستخدام الأمر (Exit) فى السطر (١٢) .

٧- السطر (١٤) إعادة تنفيذ النافذة (A) لعرض رسالة أخرى داخل النافذة واستقبال ما يدخله المستخدم داخل حقل الذاكرة المتغير الذى تم إنشاؤه فى سطر (١٥) .

٨- السطر (١٩) للتفتيش فى ملف قاعدة البيانات (Store) عن السجل الذى فيه رقم الصنف يساوى الرقم الذى أدخله المستخدم وتم تخزينه فى حقل الذاكرة المتغير (Mno) .

٩- من السطر (٢٠) الى (٢٧) حلقة شرطية تعنى أنه فى حالة عدم العثور على السجل فإنه يتم إحداث صوت الجرس مع عرض النافذة (A) وبداخلها رسالة تفيد عدم وجود هذا الصنف ، ويتم الإنتقال الى السطر (٤) بواسطة الأمر (Loop) فى السطر (٢٦).

١٠- السطر (٢٨) لحذف السجل بصفة مبدئية .

١١- من السطر (٢٩) الى (٣٣) حلقة شرطية تعنى أنه فى حالة تأكد المستخدم من الرغبة فى حذف السجل ، فإنه يتم حذفه بشكل نهائى من الملف باستخدام الأمر (Pack) فى السطر (٣٠) ، وفى حالة رغبة المستخدم فى أن لا تتم عملية الحذف ، فإنه يتم استعادة السجل باستخدام الأمر (Recall) فى السطر (٣٢) .

١٢- فى السطر (٣٤) يتم تنفيذ النافذة (A) مرة أخرى لعرض رسالة تستفسر من المستخدم عما إذا كان يريد شطب صنف آخر .

١٣- من السطر (٣٩) الى (٤٣) حلقة شرطية تعنى أنه فى حالة رغبة المستخدم فى شطب صنف آخر فإنه يتم الإنتقال الى السطر (٤) ، وفى حالة عدم رغبته فى شطب صنف آخر فإنه يتم الخروج من الحلقة الدوارة اللا نهائية باستخدام الأمر (Exit) فى السطر (٤٢) ويتم الخروج الى السطر (٤٥).

١٤- السطر (٤٥) للإنتقال الى القائمة .

المثال الثانى :

فى المثال السابق تم إنشاء القائمة الرئيسية للبرنامج باستخدام كل من القائمة الأفقية (Bar Menu) متصلة بعدد من القوائم الرأسية (Pop-Up) وهذه القوائم معرفة أصلاً فى نظام dBase IV ويقوم المستخدم بتعريفها وضبطها لتناسب أغراضه .
وفى المثال التالى برنامج يقوم بإنشاء القائمة الرئيسية يتم التحرك بين اختياراتها باستخدام الاضاءة المتحركة باستخدام أوامر dBASE .
وعند تنفيذ البرنامج تعرض الشاشة التالية :

NENU PROGRAM DEMO

AUTHER : ENG. ALY BAHLOUL

MAIN MENU

DATA	CODE
SEARCH	MAINT
HELP	EXIT

FOR DATA ENTRY AND MODIFY

نص البرنامج :

```
clear
close all
clear all
set talk off
set echo off
set status off
set escap off
set scoreboard off
set date british
select 1
use train index no
select 2
use course index num
store reccount () to acount
select 3
use job
select 1
pub1 cot
cot=0
do menu
x=10
y=22
f=1
inf=space(7)
do while .t .
s=0
@ 10,22 say " D A T A "
@ 10,55 say " C O D E "
@ 14,22 say "SEARCH"
@ 14,55 say " MAINT  "
@ 18,22 say " H E L P  "
@ 18,55 say " E X I T   "
do case
```

```
    case f=1
    inf='D A T A'
    do mess1 with " FOR DATA ENTRY AND MODIFY"
    case f=2
    inf='C O D E'
    do mess1 with "FOR EDIT CODES"
    case f=3
    inf='SEARCH'
    do mess1 with "FOR LOOK FOR DATA"
    case f=4
    inf="MAINT"
    do mess1 with     "FOR COPY DATA TO FLOPPY AND;
    RE_ INDEX FILES"
    case f=5
    inf='H E L P'
    do mess1 with "PROGRAM HELP"
    case f=6
    inf='E X I T'
    do mess1 with "TO EXIT TO DOS"
endcase
s=0
do while s=0
    s=inkey( )
    set colo to gr+ /bg+
    @ x,y say inf
    set colo to
enddo
if F=1
do case
CASE S=5
F=5
X=18
Y=22
LOOP
case s=19 . or . s=4
F=2
```

```
X=10
Y=55
LOOP
case s=24
F=3
X=14
Y=22
LOOP
case s=13
do data
endcase
endif
if F=2
 do case
case s=4 . or . s=19
F=1
X=10
Y=22
LOOP
case s=5
F=6
X=18
Y=55
LOOP
case s=24
F=4
X=14
Y=55
LOOP
case s=13
do code
endcase
endif
if f=3
do case
case s=4 . or . s=1g
F=4
```

```
X=14
Y=55
LOOP
case s=5
F=1
X=10
Y=22
LOOP
case s=24
F=5
X=18
Y=22
LOOP
case s=13
do search
endcase
endif
if F=1
do case
case s=19 . or . s=4
F=3
X=14
Y=22
LOOP
case s=5
F=2
X=10
Y=55
LOOP
case s=24
F=6
X=18
Y=55
LOOP
case s=13
do maint1
edncase
```

```
indif
if F=5
do case
case s=4 . or . s=19
F=6
X=18
Y=55
LOOP
case s=24
F=1
X=10
Y=22
LOOP
case s=5
F=3
X=14
Y=22
case s=13
clear
do demo
endcase
endif
if F=6
do case
case s=4 . or . s=19
F=5
X=18
Y=22
LOOP
case s=5
F=4
X=14
Y=55
LOOP
case s=24
F=2
X=10
```

```
Y=55
case s=13
cleaR
close data
quit
return
endcase
endif
ENDDO
DO MENU
return
*************
PROCEDURE MENU
@ 0,0  say replicate ( chr (176) , 79 ) colo bg+
@ 1,1  to 1,79 doub colo r / n
@ 2,25  say "MENU PROGRAM DEMO" colo gr+/n
@3,5  say "AUTHER : ENG. ALY BAHLOUL" colo gr+/n
@ 5,1 to 5,78 colo r / n
@ 6,27 to 8,55 colo bg+/n
@ 7,31 say "MAIN MENU" colo gr+/n
@ 9,21 to 11,29 colo bg+/n
@ 9,54 to hh,62 colo bg+/n
@ 13.21 to 15,29 colo bg+/n
@ 13,54 to 15,62 colo bg+/n
@ 17,21 to 19,29 colo bg+/n
@ 17,54 to 19,62 colo bg+/n
@ 23,1 to 23,78 colo r/n
@ 10,20 say chr (176) colo gr+
@ 10,53 say chr (176) colo gr+
@ 14,20 say chr (176) colo gr+
@ 14,53 say chr (176) colo gr+
@ 18,20 say chr (176) colo gr+
@ 18,53 say chr (176) colo gr+
@ 10,22 say "D A T A"
@ 10,55 say "C O D E"
@ 14,22 say"SEARCH"
@ 14,55 sagy"MAINT"
```

```
220 @ 18,22 say "H E L P"
221 @ 18,55 say "E X I T "
222 @  20,1 to 20,78 colo r/n
223 @  23,0 say replicate ( chr (176) , 79 ) colo bg+
224 return
225  *********************
226   PROCEDURE MESS
227 parameter mes
228 @ 23,2 SAY SPACE (77)
229 @ 23, int ( (80 - len(mes) ) / 2 ) say mes
230 return
231  **********************
232 procedure mess1
233 parameter mess1
234 @ 22,1 say space (78)
235 @ 22, int ( 80 - len (mes1) ) / 2 say mes1
236 return
```

****

شرح البرنامج :

١ - السطر (٢١) لتنفيذ الاجـراء (Menu) الذى يعرض الشـكل الأسـاسى لشاشة القائمة .
الاجراء (Menu) يبدأ من سطر (١٩٥) حتى سطر (٢٢٤) .

٢ - السطور التالية لسطر (٢١) خاصة بالحركة بين اختيارات القائمة الستة ، حيث تتم الحركة بضغط المستخدم على مفاتيح الحركة (→ ← ↑ ↓) وعند قيام المستخدم بضغط مفتاح < enter > عند أى اختيار من اختيارات القائمة يتم التفريع لتنفيذ البرنامج الفرعى ، وبعد الانتهاء من التنفيذ يتم الرجوع لعرض القائمة مرة أخرى .

٣ - من سطر (٢٢) الى سطر (٢٤) إنشاء ثلاث حقول متغيرة :
(Y , X) لتحديد موقع اختيار القائمة على الشاشة .
(F) لتحديد اختيار معين من اختيارات القائمة (من ١ الى ٦) .

٤ - السطر (٢٥) حقل متغير لتخزين عنوان الاختيار كما هو معروض على الشاشة .

٥ - من سطر (٢٦) الى السطر (١٩١) حلقة دوارة .

٦ - من سطر (٢٨) الى سطر (٣٣) لعرض اختيارات القائمة في مواقعها المحددة .

٧ - من سطر (٣٥) الى سطر (٥٤) حلقة شرطية متعددة ، تعنى أنه فى حالة اختيار أحد خيارات القائمة فإنه يتم تخصيص الرسالة التى تناسبه والتى ستعرض أسفل الشاشة .

٨ - من سطر (٥٦) الى سطر (٦١) حلقة دوارة تستخدم لاستنتاج المقابل لأحد مفاتيح الحركة باستخدام الوظيفة ( ) inkey .

٩ - سطر (٥٨) ضبط الألوان لإعطاء شكل مؤشر الإضاءة للاختيار الحالى بناءاً على المعاملات التى تعوض عن الاختيار الحالى و X , Y اللذين يحددان احداثيات الاختيار الحالى على الشاشة .

١٠ - من سطر (٦٣) وحتى سطر (١٩٠) عملية معالجة لاحتمالات ضغط المستخدم على أى من مفاتيح الحركة الأربعة باستخدام الوظيفة ( ) inkey .

مفتاح الحركة لليمين ⟵ = ٤
مفتاح الحركة لليسار ⟶ = ١٩
مفتاح الحركة لأعلى ↑ = ٥
مفتاح الحركة لأسفل ↓ = ٢٤

فمثلاً فى حالة أن يكون الاختيار الحالى هو الاختيار = "Data"حيث F=1 وتم ضغط مفتاح الحركة لليمين ، فيتم اعطاء القيمة (٢) للحقل المتغير (F) والذى يعوض عن الاختيار الثانى للقائمة وهو الاختيار (Code) ويتم الرجوع الى سطر (٢٧) كنتيجة لتنفيذ الأمر (LOOP) .

وعند قيام المستخدم بضغط مفتاح < enter > فذلك يعنى التفريع لتنفيذ أحد البرامج الفرعية .
ومقابل مفتاح < enter > تبعاً لوظيفة ( ) inkey هو ١٣ .

الفصل الخامس عشر

* أهمية ربط قاعدة البيانات ببرنامج الأتوكاد .

* إنشاء المشروعات باستخدام dBASE.

* جدوى إنشاء المشروعات باستخدام dBASE.

* إدارة المشروعات .

الفصل الخامس عشر

ربط قاعدة البيانات ببرنامج الأتوكاد

يعتبر برنامج الأتوكاد من أقوى برامج تصميم المشروعات باستخدام الحاسب .

وفى هذا الفصل أفكار مقترحة تعطى مدى واسع من المرونة فى التعامل مع برنامج الأتوكاد من خلال dBase أو التعامل مع dBase من خلال الأتوكاد .

وإمكانية تنفيذ هذه الأفكار تعتمد فقط على مهارتك فى إنشاء برامج dBase وفى الفهم الواعى بأوامر الأتوكاد .

ومستخدموا برنامج الأتوكاد المتقدمون يعلمون جيداً قصور برنامج الأتوكاد فى تنفيذ بعض المهام مثل إدارة المشروعات أو إجراء علميات حسابية مما يجعلهم يلجأون للغة الأتوليسب (Autolisp) المعقدة لإنجاز هذه المهام المكملة .

وهذا الفصل محاولة لجعل dBase قادرة على إنجاز هذه المهام بكفاءة تامة ، ولكن قد يتطلب الأمر فى بعض المهام إلى إمكانيات خاصة للحاسب مثل سعة الذاكرة وسرعة المعالج .

يمكن تقسيم المهام الممكن إنجازها باستخدام dBase مع الأتوكاد الى :

١ - إنشاء المشروعات .

٢ - إدارة المشروعات .

أولاً : إنشاء المشروعات باستخدام dBase

توجد إمكانية فى برنامج الأتوكاد تستطيع بواسطتها تجميع أى عدد من أوامر الأتوكاد وتنفيذها كلها مرة واحدة وذلك باستخدام ملف (script file) والذى له الامتداد (. scr) .

فلو استطعنا إنشاء ذلك الملف باستخدام dBase فإننا نستطيع أن ننفذه باحدى طريقتين :

١ – من داخل dBase ولكن ذلك يستلزم ألا تقل ذاكرة الحاسب عن ٢ ميجا بايت .

٢ – من داخل الأتوكاد .

وبذلك يمكنك إنشاء أى رسومات من خلال dBase عن طريق إنشاء برنامج يقوم بتوجيه خرج بعض الأوامر الى ملف يكون له الامتداد (scr .) ، وكما علمنا فإن هذا الملف يحتوى على عدد من أوامر الأتوكاد يتم تنفيذها كلها مرة واحدة .

ولفهم ذلك إليك المثال التالى :

داخل برنامج الأتوكاد :

عند الرغبة فى رسم خط مثلاً تستطيع كتابة الأمر التالى من خط الأوامر (commond line)

COMMAND : LINE

اضغط < ENTER>

فيرد الأتوكاد بالرسالة الآتية :

FROM POINT :

أكتب إحداثى نقطة البداية ولتكن (3,3) مثلاً ثم اضغط < ENTER>

فيتم عرض الرسالة التالية :

TO POINT :

اكتب إحداثى نقطة النهاية ولتكن (6,6)

ثم اضغط < ENTER>

ستعرض الرسالة التالية :

TO POINT :

اضغط <ENTER> لإنهاء التعامل مع أمر LINE

ولنفترض أننا نريد أن ننشئ ملف (script file) لتجميع هذه الخطوات ليتم تنفيذها مرة واحدة .

من خلال منقح كلمات (Editor) مناسب مثل (Norton editor) أو أى منقح كلمات آخر مشابه (لا ينصح باستخدام المنقح الخاص بـ dBase فى مثل هذه المهام) .

يتم كتابة نفس الخطوات السابقة وعند كتابة أمر معين يجب أن تكون على دراية تامة برد فعل الأتوكاد على هذا الأمر حتى تضمن التسلسل الدقيق فى تنفيذ الأوامر ، ويتم التعويض عن ضغط مفتاح <ENTER> بما يلى :

أ - إما بسطر فارغ (Blank line)

ب - أو بمسافة واحدة فارغة (Space)

وبذلك يصبح ملف (script file) محتوياً على السطر التالى :

LINE ▽3,3▽ 6,6 ▽

أو السطور التالية :

LINE

سطر فارغ

3,3

سطر فارغ

6,6

سطر فارغ

سطر فارغ

إنشاء ملف (. scr) داخل dBase

يتم ذلك باستخدام الأمر

SET ALTERNATE TO < filename >
SET ALTERNATE ON/OFF

حيث يتم إرسال خرج تنفيذ بعض الأوامر الى ملف له امتداد (. scr) .

ثم يتم نسخ هذا الملف الى دليل الأتوكاد لتنفيذه

RUN COPY < filename >.scr \ACAD

مثال : لإنشاء ملف يقوم بتنفيذ الخط السابق شرح خطوات تنفيذه سابقاً من خلال برنامج dBase

```
SET TALK OFF
SET  ALTERNATE TO SAMPLE . SCR
STORE  0 TO X1, X2, Y1, Y2
CLEAR
@ 5,5 SAY " PLEASE ENTER START X CO_ORDUCTE:";
   GET X1 PICTURE "99.99"
@ 5,5 SAY " PLEASE ENTER THE END POINT Y;
   CO_ORDINATE : ";
    GET Y1 PICTURE "99.99"
@  5,5 SAY " PLEASE ENTER THE START POINT Y;
    CO_ORDINATE: "
   GET X2 PICTURE "99.99"
@ 5,5 SAY " PLEASE ENTER THE END POINT Y
   COORDINATE:";
   GET Y2 PICTURE "99.99"
SET ALTERNATE ON
@ 0.0
?  " LINE "
??  "  "
??  LTRIM ( STR ( X1, 5,2))
??  "  "
??  LTRIM ( STR ( Y1, 5,2))
??  "  "
??  LTRIM ( STR ( X2, 5,2))
??  "  "
??  LTRIM ( STR ( Y2, 5,2))
??  "  "
CLOSE ALTERNATE
RUN COPY SAMPLE .SCR \ACAD
RETURN
```

يلاحظ فى البرنامج السابق ما يلى :

١ - يتم أولاً استقبال إحداثيات الخط السينية والصادية لنقطتى البداية والنهاية . وذلك يعنى إمكانية استخدام نفس البرنامج مرة أخرى بقيم مختلفة لهذه الإحداثيات .

٢ - يتم بعد ذلك استخدام الأمر :

SET ALTERNATE ON

ليتم توجيه خرج استخدام الأمر (?) أو (??) الى ملف (SAMPLE . SCR)

٣ - يتم أخيراً اغلاق الملف باستخدام الأمر

CLOSE ALTHERNATE

٤ - ثم يتم نسخ الملف SAMPLE .SCR الى دليل الأتوكاد حيث يمكن تنفيذه من داخل الأتوكاد .

مثال :

المثال التالى يقوم بإمكانية لإنشاء عدد من الرسومات من خلال برنامج dBase

وعند تنفيذ هذا البرنامج يتم عرض القائمة التالية :

DRAWING MENU

1 = DRAW LINE
2 = DRAW CIRCLE
3 = DRAW TREINFE
4 = DRAW POLYGON
5 = EXIT

ENTER YOUR CHOICE :

وعند قيام المستخدم باختيار أى من خيارات القائمة فإنه يتم عرض مجموعة من الرسائل تطلب تحديد احداثيات الرسم المختار ليتم إنشاء script file يحتوى على مجموعة من أوامر الاتوكاد التى تتولي تنفيذ هذا الرسم .

```
set talk off
if file ('test.scr')
delete file test. scr
endif
set alter to test.scr
@ 5,20   say " DRAWING MENU "
@ 8,10   say " 1 = DRAW LINE "
@ 10,10 say " 2 =  DRAW CIRCLE "
@ 12,10 say " 3 =  DRAW TRAINGLE "
@ 14,10 say " 4 =  DRAW POLYGON "
@ 16,10 say " 5 =  EXIT "
mch=0
@ 18,5 say " ENTER YOUR CHOICE: ";
    get  mch  picture "9" range 1,6
read
do case
case mch=1
clear
store 0 to x1, x2 , y1 , y2
@ 10,10 say " please enter the first co_ordinates of the  line(x1) : ";
    get x1 pict "99.99"
read
@ 10,10 say " please enter the first co_ordinates of the  line(y1) : ";
    get y1 pict "99.99"
read
@ 10,10 say " please enter the first co_ordinates of the  line(x2) : ";
    get x2 pict "99.99"
read
@ 10,10 say " please enter the first co_ordinates of the  line(y2) : ";
    get y2 pict "99.99"
read
set alter on

** to  draw a line

@  0,0
?? "  "
?  ltrim ( str (x1,5,2))
?? ","
?? ltrim ( str (y1,5,2))
```

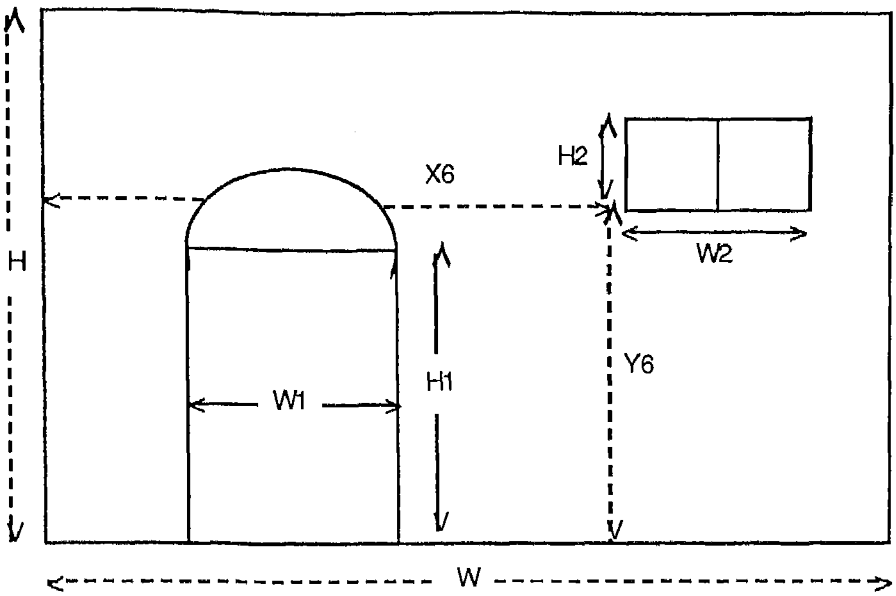
```
?? "  "
?? ltrim ( str (x2,5,2))
?? ","
?? ltrim ( str (y2,5,2))
?? " "
set alter off
clos alter

** to draw a circle

case mch=2
clear
x=0
y=0
r=0
@ 10,10 say " please enter the (X) co_ordinates of the  centre(x) : ";
   get  x  pict "99.99"
read
@ 10,10 say " please enter the (Y) co_ordinates of the  centre(y) : ";
   get  y  pict "99.99"
read
@ 10,10 say " please enter the radius of the circle        "  get  x;  pict;
   "99.99"
read
SET  ALTER  ON
@  0,0
?? "CIRCLE"
?? " "
?? " ltrim ( str ( X,5,2))
?? ","
?? ltrim ( str ( Y,5,2))
?? " "
?? ltrim ( str ( R,5,2))
?? " "
SET  ALTER  OFF
CLOS  ALTER

** to draw a triangle

case mch=3
```

```
clear
store 0 to x1 , x2 , x3 , y1 , y2 , y3
@ 10,10 say " please enter the co_ordinates (x1) : "  get  x1;
    pict "99.99"
read

@ 10,10 say " please enter the co_ordinates (y1) : "  get  y1;
    pict "99.99"
read
@ 10,10 say " please enter the co_ordinates (x2) : "  get  x2;
    pict "99.99"
read
@ 10,10 say " please enter the co_ordinates (y2) : "  get  y2;
    pict "99.99"
read
@ 10,10 say " please enter the co_ordinates (x3) : "  get  x3;
    pict "99.99"
read
@ 10,10 say " please enter the co_ordinates (y3) : "  get  y3;
    pict "99.99"
read
SET  ALTER  ON
@  0,0
??  "LINE"
??  "  "
??  ltrim ( str (x1,5,2))
??  ","
??  ltrim ( str (y1,5,2))
??  " "
??  ltrim ( str (x2,5,2))
??  ","
??  ltrim ( str (y2,5,2))
??  " "
??  ltrim ( str (x3,5,2))
??  ","
??  ltrim ( str (y3,5,2))
??  " "
??  "CLOSE"
??  "  "
SET  ALTER OFF
CLOS ALTER
```

```
**  to draw a polygon

case mch=4
clear
STORE O TO NO,X,Y,R
@ 10,10 say " please enter the no of sides of the polygon :  " get no;
   pict "99.99"
read
@ 10,10 say " please enter the (X) co_ordinate of the centre of  the;
   polygon(x) :  " get  x  pict "99.99"
read
@ 10,10 say " please enter the (Y) co_ordinate of the centre of  the;
   polygon(y) :  " get  y  pict "99.99"
read
@ 10,10 say " please enter the radious of the circle which contains;
   the polygon :  " get  y  pict "99.99"
read
SET  ALTER  ON
@  0,0
??  "POLYGON"
??  "  "
??  ltrim ( str (no,5,2))
??  " "
??  ltrim ( str (x,5,2))
??  ","
??  ltrim ( str (y,5,2))
??  " "
??  " I "
??  "  "
??  ltrim ( str (r,5,2))
??  " "
SET  ALTERNATE  OFF
CLOS  ALTERNATE
ENDCASE
RETURN
```

مثال :

نستطيع الآن أن نقوم بإنشاء رسم أكثر صعوبة .. أنظر الشكل التالى



شكل (١٥ – ١)

هل تستطيع أن تنشئ برنامجاً يمكنه إنشاء مثل هذا الرسم مع الأخذ بالاعتبار أن أبعاد هذا الرسم متغيرة ، أى أنك ستنشئ البرنامج وستدع للمستخدم إمكانية اختيار الأبعاد المناسبة له .

نص البرنامج

أكتب البرنامج التالى كما هو ثم قم بتنفيذ ملف HOUSE .SCR من خلال برنامج الأتوكاد لتحصل على الشكل الموضح بالشكل (١٥ - ١) مع الاهتمام باستخدام منقح كلمات مناسب مثل منقح نورتون ، أما إذا استخدمت منقح الكلمات الخاص بـ dBase فقد ينتج بعض المشاكل المتعلقة بالمسافات الفارغة والتى قد تسبب عدم إمكانية تنفيذه بالصورة الصحيحة .

```
set talk off
set alter to HOUSE .SCR
clear
@ 5,20 say " DRAW A HOUSE "
@ 6,20 say " ============= "
@ 10,2 say " THE OUT FRAM OF HOUSE "
   STORE O TO X1,Y1,W,H,X2,Y2,W1,H1,W2,H2,X3,Y3,X4,Y4,;
   X5,Y5,X6,Y6
@ 12,5 say " ENTER THE LOWER LEFT CORNER CO_ORDINATES
   OF THE HOUSE : "
@ 13,10 say " X-CO_ORDINATE "  GET X1 PICT "99.99"
READ
@ 13,10 say " Y-CO_ORDINATE "  GET Y1 PICT "99.99"
READ
@ 12,5 say " ENTER THE WIDTH OF THE HOUSE : " GET W ;
   PICTURE "99.99"
READ
@ 12,5 say " ENTER THE HEIGHT OF THE HOUSE : " GET H ;
   PICTURE "99.99"
READ
@ 12,5 say " ENTER THE LOWER LEFT CORNER CO_ORDINATES
   OF THE DOOR: "
@ 13,5 say " X-CO_ORDINATE : " GET X2 PICT "99.99"
READ
@ 13,10 say " Y-CO_ORDINATE : " GET Y2 PICT "99.99"
READ
@ 12,1 CLEAR
@ 12,5 say " ENTER THE WIDTH OF THE DOOR : " GET W1 ;
   PICTURE "99.99"
```

```
READ
@ 12,5 say " ENTER THE HEIGHT OF THE DOOR : " GET H1;
   PICTURE "99.99"
READ
@ 12,0 CLEAR
@ 12,5 say " ENTER THE LOWER LEFT OF WINDOW;
   CO_ORDINATES : "
@ 13,5 say " ENTER THE X-DIMENSIN FROM LOWER LEFT;
   HOUSE CORNER : " GET X6 PICTURE "99.99"
READ
@ 13,5 say " ENTER THE Y-DIMENSIN FROM LOWER LEFT;
   HOUSE CORNER : " GET Y6 PICTURE "99.99"
READ
@ 13,5 say " ENTER THE WIDTH OF THE WINDOW : " GET W2;
   PICTURE "99.99"
READ
@ 13,5 say " ENTER THE HEIGT OF THE WINDOW : "   GET H2;
   PICTURE "99.99"
READ
DH=Y2+H1
DW=X2+W1
AW=X2+W1
AH=Y2+H1
AH1=AH+1
AW1=(W1/2)+X2
XW=X2+X6
YW=Y2+Y6
XMW=XW+(W2/2)
YMW=YW+H2
CLEAR
@  0,0
SET  ALTERNATE  ON
??  " LINE "
??  "  "
?? LTRIM ( STR (X1,5,2))
?? ","
?? LTRIM ( STR (Y1,5,2))
?? " "
?? "@"
?? LTRIM ( STR (W,5,2))
```

```
?? ","
?? LTRIM ( STR (0,1,0))
?? " "
?? "@"
?? LTRIM ( STR (0,1,0))
?? ","
?? LTRIM ( STR (H,5,2))
?? " "
?? "@"
?? LTRIM ( STR (-W,5,2))
?? ","
?? LTRIM ( STR (0,1,0))
?? "  "
?? "C"
?? "  "
? " LINE "
?? "  "
?? LTRIM ( STR (X2,5,2))
?? ","
?? LTRIM ( STR (Y2,5,2))
?? "  "
?? LTRIM ( STR (X2,5,2))
?? ","
?? LTRIM ( STR (DH,5,2))
?? "  "
?? "@"
?? LTRIM ( STR (W1,5,2))
?? ","
?? LTRIM ( STR (0,1,0))
?? "  "
?? "@"
?? LTRIM ( STR (0,1,0))
?? ","
?? LTRIM ( STR (-H1,5,2))
?? "  "
?? " C "
?? "ARC"
?? "  "
?? LTRIM ( STR (AW,5,2))
?? ","
```

```
?? LTRIM ( STR (AH,5,2))
?? " "
?? LTRIM ( STR (AW1,5,2))
?? ","
?? LTRIM ( STR (AH1,5,2))
?? " "
?? LTRIM ( STR (X2,5,2))
?? ","
?? LTRIM ( STR (AH,5,2))
?? "LINE"
?? " "
?? LTRIM ( STR (XW,5,2))
?? ","
?? LTRIM ( STR (YW,5,2))
?? " "
?? "@"
?? LTRIM ( STR (W2,5,2))
?? ","
?? LTRIM ( STR (0,1,0))
?? " "
?? "@"
?? LTRIM ( STR (0,1,0))
?? ","
?? LTRIM ( STR (H2,5,2))
?? " "
?? "@"
?? LTRIM ( STR (-W2,5,2))
?? ","
?? LTRIM ( STR (0,1,0))
?? " "
?? "C"
? "LINE"
?? " "
?? LTRIM ( STR (XMW,5,2))
?? ","
?? LTRIM ( STR (YW,5,2))
?? " "
?? "@"
?? LTRIM ( STR (0,1,0))
?? ","
```

```
?? LTRIM ( STR (H2,5,2))
?? " "
CLOSE  ALTERNATE
RUN COPY HOUSE.SCR   \ ACAD
RETURN
```

جدوى إنشاء مشاريع باستخدام dBase

ما جدوى إنشاء مشروع رسم باستخدام dBase ما دام يمكننا القيام بهذه العملية مباشرة باستخدام الأتوكاد ؟

ربما خطر ببالك هذا السؤال ..

ويمكننا تلخيص الأجابة على هذا التساؤل فى هذه النقاط :

١ – فى حالة تكرار رسم نفس المشروع بأبعاد أو بمواصفات متغيرة .

٢ – فى حالة الحاجة لرسم مشروع يحتاج لعمليات حسابية كبيرة يكون من الأسهل اجراؤها باستخدام برنامج dBase حيث أن إمكانية الحساب غير متاحة داخل الأتوكاد .

٣ – فى حالة الحاجة لرسم مشروع به تحفظات خاصة بأبعاد الرسم أو بتجانس مكونات الرسم أو بتناسب مكونات الرسم بالنسبة للرسم ككل .

٤ – يمكن بعد اكتساب الخبرة الكافية إنشاء مشاريع كاملة بكتابتها على الورق ليقوم بعد ذلك أى مستخدم عادى بتنفيذها على الحاسب دون الحاجة لخبرة سابقة فى الرسم بحيث يمكن لهذا المستخدم كذلك أن يطوع هذا المشروع لمتطلباته .

وإذا قمنا بتطبيق هذه النقاط على المثال الأخير لأمكننا ملاحظة الآتى :

١ – كما هو واضح يمكن إعادة رسم المنزل بأبعاد مختلفة كل مرة .
٢ – أمكن استنتاج أبعاد بعض المكونات مثل خط المنتصف فى النافذة أو القوس أعلى الباب .
٣ – من الممكن عمل مواصفات خاصة بأبعاد الرسم مثل نسبة طول وعرض النافذة بالنسبة لأبعاد المنزل أو أبعاد الباب بالنسبة للمنزل .. وهكذا ..
٤ – يتضح أنه يمكن كتابة البرنامج كاملاً على الورق شاملاً المواصفات أو التحفظات الخاصة بأبعاد الرسم ثم إنشاء حقول ذاكرة متغيرة لاستقبال أبعاد الرسم من المستخدم ثم التعويض بها أثناء كتابة أوامر الأتوكاد .

ثانياً : إدارة المشروعات
باستخدام dBase

يحتـــوى برنامــــج الأتوكـــــاد على قاعـــدة بيـانات خاصـــة به وهى عبـارة عن مجمـوعة (Attributes) المستخدمة فى المشروع .

ويمكن استغلال هذه الامكانية فى تسجيل بيانات مكونات المشروع ، وذلك بارسال بيانات قاعدة البيانات هذه (والتى هى فى الأصل بيانات الـ (Attribates) الى dBase حيث يمكننا بسهولة معالجة هذه البيانات ومتابعة التحديث فيها واستخراج النتائج والتقارير أولاً بأول .

ولتقريب ذلك من أذهاننا إليك المثال الآتى :

لنفترض أنه فى هيئة الأرصاد الجوية يتم التنبؤ بحالة الطقس نتيجة لمجموعة من العوامل مثل درجة الحرارة والضغط والرياح والرطوبة فى بقاع العالم المختلفة .

ولنفترض أننا نريد أن ننشئ مشروع باستخدام كل من الأتوكاد و dBase للتحكم فى هذه العملية .

يتم ذلك ببساطة برسم خريطة للعالم باستخدام الأتوكاد وذلك بشفها (Digitizing) باستخدام (Digitizer) ويتم عمل (Block) يحتوى على مجموعة من (Attrilutes) مثل [الموقع – درجة الحرارة – الضغط – الرياح – الرطوبة] لكل بقعة معينة على الخريطة .

ويتم التحديث اليومى على هذه الـ (Attributes) لكل بقعة على الخريطة ثم يتم نسخ هذه البيانات الى dBase حيث يتم تحليلها تبعاً لقوانين الأرصاد ليتم فى النهاية استنتاج تقرير باحتمالات الطقس لليوم التالى أو للأيام التالية .

خطوات إنشاء المشروع :

١ - يتم أولاً رسم خريطة للعالم باستخدام Difitizer

٢ - يتم عمل Block يحتوى على Attribuite الآتية :

1 - LOCATION
2 - TEMPRATURE
3 - PRSSURE
4 - WIND$

٣ - يتم عمل حشر (insert) لهذا (block) فى مواقع معينة على الخريطة .

٤ - يتم إنشاء الملف الوسيط باستخدام (Templete file) منقح الكلمات كما يلى :

LOCATION	C020000
TEMMPRATURE	N005002
PRESSURE	N005002
WINDS	C020000

ثم يتم استنتاج ملف نص (text file) يحتوى على بيانات (Attributes) هذه باستخدام أمر الأتوكاد (ATTEXT) ويتم نسخ هذا الملف الى دليل dBase
ولإمكانية نسخ هذه البيانات الى ملف قاعدة بيانات يتم الآتى :
١ - يتم إنشاء ملف قاعدة بيانات تكون بنيته (structures) لها نفس أسماء (Attributes) المستخدمة فى الأتوكاد ونفس النوع (إن كانت حرفية أو عددية) ونفس الطول كما هو موضح فى الملف الوسيط (tamplete file)
ثم إنشاء برنامج يتحتوى على السطرين التاليين :

```
USE < filename >
APPEND FROM < filename > . TXT SDF
```

وبذلك يكون لدينا كل بيانات قاعدة بيانات الرسم ونستطيع أن نعالجها ونستخرج منها التقارير المختلفة بسهولة تامة .

الفصل السادس عاشر

ملخص أوامر dBaseIV

الفصل السادس عاشر

ملخص أوامر dBaseIV

1 - ?/?? [< exp1 > [picture < exp c] [Function < function list >]
[AT < exp N >] [STYLE < font number >]]

تستخدم لعرض أو طبع تعبير أو أكثر .

? للعرض أو الطبع من بداية السطر التالي .

?? للعرض أو الطبع مباشرة بعد الجملة السابق عرضها أو طبعها .

Picture لضبط شكل ظهور التعبير .

Function تستخدم لتحديد شكل البيانات المعروضة أو المطبوعة .

Style لطبع أو اظهار تعبير بأكثر من طراز مثل الخط المائل italic أو رسم خط تحت التعبير underline .. الخ .

AT لتحديد موقع طبع أو عرض التعبير .

2 - ??? < exp c >

تستخدم لطباعة تعبير أو أكثر بغض النظر عن نوع مشغل (driver) الطابعة .

3 - @ ..... say ..... Get

```
@ < row > , < col > [ say < exp > [ Picture < exp c > ]

                   [ Function < function list > ] ]
                   [  [ open ] Window  < wind. name > ]
              [ Get  < Variable > ]
                   [PICTURE < fexp c > ]
                   [ Range < low > ] , < high > ]
                   [ VALID  <  condition  > ]
                   [ When < condition > ]
                   [ Default < exp > ]
                   [ MESSAGE < exp c > ]
                   [ COLOR [ < standard > ]
                   [ , < enhanced ] ]
```

١ - a < r > , < c > say

تستخدم لاظهار تعبير على الشاشة .

٢ - Get < var >

تتيح اظهار وتعديل حقول ملف قاعدة البيانات أو حقول ذاكرة متغيرة .

٣ - Picture < exp c >

تستخدم لضبط شكل البيانات التى تعرض أو التى تستقبل من المستخدم .

٤ - Function < funct. list >

تستخدم لتحديد شكل البيانات المعروضة أو المستقبلة من المستخدم .

٥ - Range < low > , < high >

تستخدم لتحديد أقل قيمة وأقصى قيمة للبيانات المستقبلة من المستخدم ، وعند ادخال المستخدم لبيانات أكبر أو أقل من المدى المحدد يعرض رسالة (outrange value) ولا يتم قبول هذه البيانات .

٦ - VALID < cond >

تستخدم لتحديد شرط يجب أن يتحقق فى البيانات المدخلة بواسطة المستخدم ، وفى حالة عدم تحقق الشرط لا يتم قبول البيانات المدخلة ، ويتم عرض الرسالة التالية :

Editing conditon not satisfied

٧ - When < cond >

تستخدم لعدم إتاحة إمكانية التعديل فى حقل ما إلا فى حالة تحقق الشرط المعطى وإلا تم إهمال هذا الحقل وتخطاه المؤشر للحقل التالى .

٨ - Default < exp >

تستخدم لإعطاء قيمة مسبقة لحقل ما ، وفى حالة الرغبة فى قبول هذه القيمة ، يتم الضغط على < enter > ، وإلا فيتم التعديل فى الحقل كالمعتاد .

٩ - Message < exp c >
تستخدم لكتابة رسالة لا تزيد عن ٧٩ حرف أسفل الشاشة .

١٠ - [OPEN] WINDOW < window name >
تستخدم لتنفيذ نافذة سبق تعريفها ، وذلك للكتابة أو التعديل فى حقل الذاكرة (Menu field) ولفتح النافذة اضغط < CTRL - HOME > .

١١ - [OLOR [< stand. >] [< enhanced >]
standard تستخدم لتحديد لون عرض < say < exp
enhanced تستخدم لتحديد لون < Get < var

4 - a < r1 > , < c1 > chear [TO < r2 > , < c2 >]

١ - @ < r > , < c >
لمسح سطر من الشاشة ابتداءاً من الإحداثى (r , c) لنهاية الخط .

٢ - @ < r > , < c > clear
لمسح مستطيل من الشاشة يكون ركنه الأعلى الأيسر الإحداثى (r , c) ، وركنه الأيمن الأسفل الركن الأيمن الأسفل من الشاشة .

٣ - @ < r1 > , < c1 > clear TO < r2 > , < c2 >
لمسح مستطيل من الشاشة يكون ركنه الأعلى الأيسر الإحداثى (r1 , c1) ، وركنه الأيمن الأسفل الإحداثى (r2 , c2) .

5 - @ ..... Fill
@ < r1 > , < c1 > TO < r2 > , < c2 >
[COLOR < color attribite >]

ينتج تغيير لون مستطيل على الشاشة .

6 - @ ..... TO

@ < r1 > , < c1 > TO < r2 > , < c2 >
[DOUBLE / PANEL / < border definition string >] [COLOR
< color attribite >]

١ - @ < r1 > , < c1 > TO < r2 > , < c2 >

لرسم مستطيل على الشاشة إحداثياته كما هو موضح .

٢ - @ < r1 > , < c1 > TO < r1 > , < c2 >

لرسم خط أفقى على الشاشة .

٣ - @ < r1 > , < c1 > TO < r2 > , < c1 >

لرسم خط رأسى على الشاشة .

٤ - DOUBLE تستخدم لتكون أحرف الخطوط مزدوجة .

٥ - PANEL تستخدم لتغيير لون المستطيل باللون العكسى .

7 - Accept

Accept [< prompt > TO < memvar >

تنشئ حقل ذاكرة متغير حرفى ، وتقوم بتخزين ما يدخله المستخدم داخل هذا الحقل .

8 - Activat Menu < menu name > [PAD < pad name >]

تنفذ القائمة الأفقية السابق تعريفها .

9 - Activat Popup < popup name >

تنفذ القائمة الرأسية السابق تعريفها .

10 - Activate screen

تستخدم للخروج من تأثير النافذة ، والتحرك بحرية خلال الشاشة دون إزالة النافذة من الشاشة .

11 - Activate window

Activate window < window name list > / ALL

تستخدم لتنفيذ النافذة السابق تعريفها حيث يتم استعراض المخرج الواصل إلى الشاشة من خلال هذه النافذة .

ALL لتنفيذ جميع النوافذ السابق تعريفها .

12 - APPEND [BLANK]

لاضافة سجل جديد فى نهاية ملف قاعدة البيانات .

BLANK ←—— لاضافة سجل جديد فى نهاية الملف ، والعودة لنقطة الأوامر أو للسطر التالى فى البرنامج لتنفيذه مع ملاحظة أن مؤشر السجلات (Pointer) يظل فى السجل الجديد المضاف .

13 - APPEND FROM

Append From < filename > / ? [[TYPE] < file type >]
[FOR < condition >]

Append from < filename >

لاضافة سجلات من ملف قاعدة بيانات الى ملف قاعدة بيانات آخر مفتوح .

اختيارات (TYPE) هى :

FW2 - DELIMITDED - DBASE II - SDF - WKS - SYLK - DIF - RPD

14 - APPEND FROM ARRAY
Append From Array < array name > [FOR < cond >]

تستخدم لاضافة سجلات من مصفوفة للف قاعدة البيانات .

15 - APPEND MEMO
Append Memo < memo file name > [FORM
< filename > [OVER WRITE]

تستخدم لاستقبال بيانات من ملف نص (text file) فى حقل ذاكرة فى سجل ملف قاعدة البيانات .

16 - ASSIST

للوصول الى مركز التحكم من نقطة الأوامر .

17 - AVERAGE
Average [< exp N list >] [< scope >] [FOR < cond >]
[WHILE < cond > [TO < memar list > / TO ARRAY
< array name >]

تستخدم لحساب المتوسط الحسابى لتعبير رقمى :

١ - Average

لحساب المتوسط الحسابى لكل الحقول العددية فى كل سجل فى ملف قاعدة البيانات .

٢ - < exp N >

تستخدم لحساب المتوسط الحسابى للحقول العددية المحددة فقط فى < exp N list >

٣ – < scop >

تستخدم لتحديد السجلات التى سوف يشملها تنفيذ الأمر .

٤ – FOR < cond >

تستخدم لترشيح سجلات معينة (التى تحقق الشرط) التى سوف يشملها تنفيذ الأمر .

٥ – WHILE < cond >

تستخدم لتنفيذ الأمر على السجلات المرشحة طالما كانت هذه السجلات تحقق شرط معين .

٦ – TO < memvar list > / To array < array name >

تخزن نتيجة تنفيذ الأمر فى حقل ذاكرة متغير أو مصفوفة .

18 - Begin Transaction / End Transaction
Begin Transaction [< path name >] commands
End transaction

هذا الأمر يستخدم على أحد امكانيتين هما :

١ – اعطاء الفرصة لالغاء التعديلات أثناء عملية التعديل واسترجاع بيانات ملف قاعدة البيانات كما كانت قبل عملية التعديل باستخدام الأمر ROLLBACK

٢ – تنفيذ التعديلات المستحدثة مع ملف قاعدة البيانات .

19 - BROWSE

تستخدم للوصول لشاشة BROWSE التى تتيح التعديل فى سجلات ملف قاعدة البيانات أو اضافة سجلات جديدة .

20 - CALCULATE

تستخدم فى حسابات البيانات المالية أو الاحصائية .

Calculate [scope] < option list > [FOR < condition >
[WHILE < condition >] [TO < memver list > / TO AR-
RAY < carray name >]

١ - [scope]

تستخدم لتحديد عدد السجلات التى سوف يشملها تنفيذ الأمر .

٢- [option list]

لتحديد وظيفة أو أكثر تستخدم مع الأمر calculate
وهذه الوظيفة أحد الوظائف التالية

MAX ( ) - CNT ( ) - AVG ( ) - VAR ( ) - SUM ( ) -
STD ( ) - NPV ( ) - MIN ( )

٣ - FOR < cond >

لتحديد السجلات التى يسوف يشملها تنفيذ الأمر والتى تحقق الشرط المحدد .

٤ - WHILE < cond. >

تستخدم لتنفيذ الأمر على السجلات المرشحة طالما كانت هذه السجلات
تحقق شرط معين .

٥ - TO < memver list > / .. To < array name >

تخزن نتيجة تنفيذ الأمر فى حقول ذاكرة مؤقتة أو فى مصفوفة

21 - CALL

call < modul name > [WITH < veriable >]

يستخدم لتنفيذ برنامج ثنائى (binary program) سبق تحميله فى
الذاكرة .

22 - Cancel

تستخدم لانهاء تنفيذ برنامج وتغلق جميع ملفات البرنامج ويتم الانتقال الى نقطة الأوامر

23 - Change

تستخدم لتغيير محتويات سجل فى ملف قاعدة بيانات مفتوح .

24 - Clear

تمسح الشاشة وتعيد وضع المؤشر فى موقعه المعتاد فى أسفل يسار الشاشة

Clear [ALL / Fields / Gets / Momory / Memus / Popups / Typeahead / Windows

١ - Clear
تمسح الشاشة وتزيل

٢ - Clear all
تغلق ملفات الكتالوج- وتغلق كل ملفات قاعدة البيانات وتزيل كل حقول الذاكرة المتغيرة ومصفوفات الذاكرة وتحذف تعريفات القوائم الأفقية والرأسية .

٣ - Clear Fields
تزيل نتيجة استخدام الأمر SET FIELDS

٤ - Clear Gets
تزيل كل أوامر GET ..... @

٥ - Clear Memory
تمسح كل حقول الذاكرة المتغيرة والمصفوفات من الذاكرة .

٦ - Clear Menus

تزيل كل القوائم الأفقية من الشاشة ومن الذاكرة .

٧ - Clear Popups

تزيل كل القوائم الرأسية من الشاشة ومن الذاكرة .

٨ - Clear Typeahead

تمسح الوسيط الخازن الخاص بعملية الكتابة .

٩ - Clear Windows

تزيل كل النوافذ من الشاشة ومن الذاكرة .

25 - Close

تستخدم لغلق ملفات قاعدة البيانات .

Close all / alternate / Database / Format / Index / Procedure

١ - Close all

تغلق جميع الملفات .

٢ - Close alternate

تغلق جميع ملفات (alternate database file) .

٣ - Close database

تغلق جميع ملفات قاعدة البيانات وكل الفهارس وملفات الذاكرة وكل ملفات التشكيل .

٤ - Close format

تغلق جميع ملفات التشكيل .

٥ – Close index

تغلق جميع ملفات الفهرسة .

٦ – Close procedur

تغلق ملف الاجراءات .

26 - Compile

تستخدم لانشاء ملف (object file) وهو ملف ناشئ عن ملف البرنامج الذى تم كتابته باستخدام أوامر dBses وقد تم ترجمتـــــه الى لغة الماكينة (machin lang.)

27 - Continue

يستخدم للبحث عن السجل التالى فى ملف قاعدة البيانات المفتوح والذى يحقق نفس الشرط السابق تحديده مع أخر أمر Locate تم استخدامه .

28 - Copy

يستخدم لنسخ ملف قاعدة بيانات مفتوح أو جزء منه .

Copy to < filename > [TYPE] < file type >] [fields list] [< scope >] [FOR < condition >] [While < condition >]

١ – اختيارات [TYPE] هى :

[DBASE II - Delimited - DIF - RPD - FW2 - delimited with < delimiter > - Delimited with Blank - WKS - SYLK - SDF]

٢ – Fields < field list >

لتحديد الحقول التى سيتم نسخها للملف الجديد .

٣ – < scop >

تحدد عدد السجلات التى سوف يتم نسخها .

٤ - FOR < condition >

تستخدم لتحديد سجلات معينة (التى تحقق الشرط) والتى سوف يتم نسخها.

٥ - While < condition >

تستخدم لمواصلة عملية نسخ السجلات طالما كانت تحقق شرط معين .

29 - Copy File

Copy File < filename > TO < filename >

يستخدم لانشاء نسخة من أى ملف .

ولا يستخدم هذا الأمر لنسخ ملف مفتوح ، ولنسخ ملف مفتوح يستخدم الأمر Copy السابق شرحه .

30 - Copy Indexes

Copy indexes < .ndx file list > TO < .mdx filename >

يستخدم لتحويل الفهارس القديمة من النوع (.ndx) الى النوع (.mdx) .

31 - Copy Memo

تستخدم لنسخ حقل ذاكرة من سجل معين فى ملف قاعدة بيانات مفتوح الى ملف نصى (Text file) .

Copy Memo To <name field name > To < filename >
[ADDITIVE]

فى حالة عدم اضافة الاختيار (ADDITIVE) يتم استبدال محتويات الملف النصى (Text file) بمحتويات حقل الذاكرة .

وفى حالة اضافة الاختيار (ADDITIVE) يتم اضافة محتويات حقل الذاكرة الى محتويات الملف النصى (Text file) .

32 - Copy Structure

يستخدم لانشاء ملف قاعدة بيانات جديد له نفس مواصفات بنيه (Structures) ملف قاعدة بيانات مفتوح .

Copy Structure To < filename > [FIELDS < field list >]

FIELDS يستخدم لتحديد حقول معينة (field list) يتكون منها الملف الجديد .

ملحوظة : لا يتم نسخ السجلات الى الملف الجديد .

33 - Copy To Structure extended

يستخدم لانشاء ملف قاعدة بيانات تحتوى سجلاته على مواصفات بنيه ملف قاعدة بيانات مفتوح .

34 - Copy Tag

يستخدم لتحويل الفهارس من النوع (mdx .) الى النوع القديم المستخدم في +dBase3 وهو (ndx.) .

35 - Copy To Array

يستخدم لنسخ محتويات سجل أو أكثر من ملف قاعدة بيانات مفتوح الى مصفوفة ذاكرة سبق تعريفها .

Copy To Array < array name > [Fields < fields lisst >]
[< scope >] [For < condition >] [While < condition >]

36 - Count

تستنتج عدد السجلات من ملف قاعدة بيانات مفتوح والتى تقابل شرط معين .

Count [To < memver >] [< scope >] [For < condition >]
[While < condition >]

37 - Create / Modify structure

تتيح الوصول الى شاشة تصميم قاعدة البيانات لانشاء ملف جديد أو للتعديل فى مواصفات بنيه ملف قديم .

38 - Create / Modify Application < filename > / ?

تتيح الوصول الى منتج التطبيقات إما لانشاء تطبيق جديد أو للتعديل فى تطبيق قديم .

39 - Create From
Create < filename > From < structure extended file >

يستخدم لانشاء ملف قاعدة بيانات جديد من مواصفات بنيه ملف سبق إنشاؤه باستخدام الأمر :

Copy Structure extended

40 - Create / Modify Label < filename > / ?

تتيح لك الوصول الى شاشة تصميم العناوين إما لإنشاء ملف عناوين جديد أو لتعديل ملف عناوين قديم .

41 - Create / Modify Query / view < filename > / ?

تتيح لك الوصول الى شاشة تصميم الاستفسارات إما لانشاء ملف استفسار جديد أو للتعديل فى ملف موجود فعلاً .

42 - Create / Modify Report < filename > / ?

تتيح الوصول الى شاشة تصميم التقارير إما لانشاء ملف تقرير جديد أو للتعديل فى ملف موجود فعلاً .

43 - Create / Modify Screen < filename > / ?

تتيح لك الوصول الى شاشة تصميم نماذج أشكال الشاشة إما لانشاء ملف جديد أو للتعديل فى ملف موجود فعلاً .

44 - Create view < . vue filename > From enviroment

تنشئ ملف عرض استفسار متوافق مع dBase III .

45 - Deactivate Menu

تزيل القائمة الأفقية من الشاشة فقط ولا تزيلها من الذاكرة .

46 - Deactivate Popup

تزيل القائمة الرأسية من الشاشة فقط ولا تزيلها من الذاكرة .

47 - Deactivate Window < window name list > ALL

تزيل النافذة أو النوافذ المحددة من الشاشة ولا تزيلهم من الذاكرة .

48 - Debug < filename > / < procedure name > [WITH
< parameter list >]

تتيح الوصول الى كاشف الأخطاء (debugger) الخاص بـ dBase IV .

49 - Declare

تســتخدم لتعريف مصــفوفات الذاكـرة والمصفوفة قد تكون أحادية الأبعاد (One dimensional) أو ثنائية الأبعاد (Two dimensional) .

50 - Define Bar

Define bar < line number > OF < popup name > prompt < exp c > [Message < exp c >] [skip [FOR < condintion >]]

تستخدم لتعريف أحد اختيارات قائمة رأسية .

51 - Define Box

Define Box From < print column > TO < print column > Height < exp N > [AT line < print line >] [Single / Double / < border definition string >]

يعرض مستطيل ليتم طباعته خلال التقرير فى الموقع المحدد .

١ - Define Box From < print col > TO < print col > Height < exp N >

لتعريف مستطيل تتم طباعته بين العموين المحددين ويكون ارتفاعه كما هو محدد .

٢ - AT line < print line >

لتحديد بدء طباعة قمة المستطيل .

٣ - Single / Double / < border def. string >

لتحديد نوع أحرف (border) المستطيل الذى سيتم طباعته .

52 - Define Menu < menu name > [Message < exp c >]

تستخدم مع الأمر (define pad) لتعريف قائمة أفقية .

53 - Define PAD

تستخدم لتعريف أحد اختيارات القائمة الأفقية .

Define PAD < pad name > OF < menu name >
PROMPT < exp c > [AT < row > , < col >] [MESSAGE
< exp c >]

54 - Define POPUP

تستخدم لتعريف القائمة الرأسية .

Define POPUP < popup > FROM < r1 > , < c1 > [TO
< r2 > , < c2 >] [PROMPT FILES [like < skelton >]]
[MESSAGE < exp c >]

55 - Define WINDOW

تستخدم لتعريف القائمة الرأسية .

Define Window < window name > FROM < r1> ,< c1>
TO < r2 > , < c2 > [DOVBLE / PANEL / NONE /
< border definition string >]
[COLOR < standard >[,< enhanced > [,< Frame >]]]

56 - Delete

يستخدم فى تمييز أو تعليم السجلات المراد حذفها .

Delete [< scop >] [FOR < cond.>] [WHILE < cond.>]
[FILE < filename >]

57 - Delete TAG

يستخدم لازالة فهرس من ملف الفهرسة المتعددة (.mdx) أو يغلق فهرس من النوع القديم (.ndx) .

Delete TAG < tag name1 > [OF < .MDX filename >] / < .NDX filename1 > ..

58 - Dir

يستخدم لعرض بيانات الدليل الحالى .

DIRECTORY / DIR / [[ON < drive >:] [LIKE] < path >] [< skelton >]

59 - DISPLAY

يستخدم لعرض أو طبع سجل أو أكثر .

DISPLAY [[FIELDS] < exp list > [OFF] [< scop >] [FOR < cond >] [WHILE < cond >] [TO PRINTER / TO FILE < filename >]

60 - DISPLAY FILES

تستخدم لعرض بيانات الدليل الحالى .

DISPLAY FILES [LIKE < skeleton >] [TO PRINTER / TO FILE < filename >]

61 - DISPLAY HISTORY

يستخدم لعرض كشف بالأوامر التى تم تنفيذها والتى خزنت فى الوسيط الخازن .

DISPLAY HISTORY [LAST < exp N >] [TO PRINTER / TO FILE < filename >]

أ – DISPLAY HISTORY

يعرض محتويات الوسيط الخازن (DISPLAY HISTORY) على الشاشة.

ب – DISPLAY HISTORY LAST

يعرض عدد آخر من أوامر سبق تنفيذها .

62 - DISPLAY MEMORY

يستخدم لعرض كل حقول الذاكرة المتغيرة .

DISPLAY MEMORY [TO Printer / TO File < filename >]

أ – DISPLAY MEMORY

يعرض كل حقول الذاكرة على الشاشة .

ب – TO PRINTER

لطباعة كل حقول الذاكرة على الطابعة .

ج – TO FILE

لتوجيه نتيجة تنفيذ الأمر الى ملف .

63 - DISPLAY STATUS

يعرض معلومات عن الحالة الحالية لقاعدة البيانات مثل رقم منطقة الذاكرة الحالية (work area) وملف قاعدة البيانات المفتوح وموقعه ، وكذلك الفهارس وملفات الذاكرة المفتوحة وملفات التشكيل (format file) .

DISPLAY STATUS [TO Printer / TO File < filename >]

64 - DISPLAY STRUCTURE

تستخدم لعرض بنيه ملف قاعدة البيانات المفتوح .

DISPLAY STRUCTURE [IN < alias name >] [TO Printer / /
TO File < filename >]

65 - DO

يستخدم لتنفيذ برنامج أو اجراء سبق تحميله فى الذاكرة .

```
DO < program filename > / < procedure name >
[ WITH < peramete list > ]
```

66 - DO CASE

يستخدم لتنفيذ فعل من عدة أفعال أو لتنفيذ فرع من عدة فروع .

```
DO < program filename > / < procedure name >
[ WITH < peramete list > ]

DO CASE
      case < cond >
         < commands >
      case < conclition >
         < commands >
      otheruwiae
         < commands >
      ENDCASE
```

67 - DO WHILE

يستخدم لتكرار الأوامر الواقعة بين do while < cond > و enddo طالما الشرط المحدد صحيح .

```
DO WHILE < cond >
      < commands >
      [ LOOP ] [ EXIT ]
ENDDO
```

68 - EDIT

تستخدم للوصول الى شاشة EDIT التى تستخدم لعرض أو تعديل البيانات فى سجلات ملف قاعدة بيانات مفتوح وكل شاشة تمثل سجل فى الملف .

```
EDIT
      NO INIT
      NO FOLLOW
      NO APPEND
      NO MENU
      NO DELETE
      NO CLEAR
         < record number >
      FIELDS < field list >
         < scope >
      FOR < condition >
      WHILE < condition >
```

69 - EJECT

يستخدم لدفع الورق فى الطابعة لقمة الصفحة التالية .

70 - EJECT PAGE

يختلف عن الأمر (EJECT) فى أنه يمكن كل وسائل استقبال الخرج (output device) فى التعامل مع الأمر (?) أى إنه مثلاً فى حالة set console on فإنه يتم توجيه الخرج (output) الى الشاشة .

71 - ERASE

يستخدم لحذف ملف من القرص الممغنط .

```
ERASE < filename >
```

72 - EXPORT

يستخدم لنسخ ملفات قاعدة بيانات تنتمى الى dBase IV الى ملفات متاحة لكل من - (FRAMLWORK - dBASE II - PFS : FILE - RAPID FILE - LOUTS)

```
EXPORT TO < filename > [ TYPE ] PFS / DBASE II / FW2 / RPD
        [ FIELDS < field list > ] [ < scop > ]
        [ FOR < condtiion > ] [ While < condition > ]
```

73 - FIND

يستخدم للبحث فى ملف قاعدة البيانات المفهرس عن أول سجل يحقق شرط البحث .

```
FIND < literal  key >
```

74 - FUNCTION

تستخدم لتعريف وظيفة جديدة غير موجودة أصلاً ضمن وظائف حيث تعطى نتيجة للبرنامج المستدعى

```
FUNCTION < procedure name >
[ PARAMETERS < parameter list > ]
< commands >
RETURN ( [ < returned value > ] )
```

75 - GO / GOTO

يستخدم لتحريك مؤشر السجلات خلال ملف قاعدة البيانات .

```
GO / GOTO BOTTOM / TOP [ IN < alies name > ]
```

76 - HELP

يستخدم لعرض معلومات إيضاحية عن أمر معين أو وظيفة معينة أو لعرض شاشة المساعدة (help screen) .

```
HELP [ < dBase IV keyword > ]
```

77 - IF

يستخدم للتحقق من صحة تعبير معين أو من عدم صحته .

IF < condition >
< commands >
ELSE
< commands >
ENDIF

78 - IMPORT

يستخدم لانشاء ملف قاعدة بيانات من ملف آخر ينتمى الى
[PFS: FILE, DBASE II, FRAMEWORK II, RAPID FILE, LOTUS - 123]

الشكل العام :

IMPORT FROM < filename > [TYPE] PFS / DBASE II / FW2 / RPD / WK1

79 - INDEX

يستخدم لانشاء ملف يقوم بفهرسة ملف قاعدة بيانات حرفياً أو عددياً .

80 - INPUT

يستخدم لحث المستخدم لادخال تعبير ويقوم بتخزينه داخل حقل ذاكرة يتم إنشاؤه .

INPUT [< prompt >] TO < memvar >

81 - INSERT

يستخدم لاضافة سجل جديد فى ملف قاعدة بيانات مفتوح عند موقع مؤشر السجلات الحالى .

INSERT [BEFOR] TO [BLANK]

أ - INSERT

يستخدم لاضافة سجل جديد مباشرة بعد مؤشر السجلات فى ملف قاعدة البيانات المفتوح ويقوم كذلك بعرض احدى شاشتى الادخال لادخال البيانات .

ب - INSERT BEFORE

يستخدم لاضافة سجل جديد عند موقع مؤشر السجلات فى ملف قاعدة البيانات المفتـــوح ويقـــوم كذلك بعرض احدى شـــــاشتى الادخـــال (Edit أو Browse) لادخال بيانات السجل .

ج - INSERT [BEFORE] BLANK

يستخدم لاضافة سجل جديد فى ملف قاعدة البيانات دون عرض احدى شاشتى الادخال .

82 - JOIN

يستخدم لانشاء ملف قاعدة بيانات جديد بحقول مختارة من ملفى قاعدة بيانات مفتوحين .

JOIN WITH < alies > TO < filename > [FIELDS < field list >] FOR < condition >

83 - LABEL FORM

يستخدم لانشاء أو لطباعة أو عرض ملف عناوين .

LABEL FORM < label filename > / [SAMMPLE]
[< scop >] [FOR < cond >] [While < cond >]
[TO PRINTER / TO FILE < filename >

أ - LABEL FORM

لانشاء أو عرض أو طبع ملف عناوين .

ب - LABEL FORM SAMPLE

لطبع عينة من العناوين لاختيار ضبط الطابعة .

84 - LIST

مرادف للأمر DISPLAY السابق شرحه إلا أنه فى حالة استخدام الأمر LIST لا يتم التوقف بعد كل شاشة لعرض البيانات .

85 - LOAD

يستخدم لتحميل ملف ثنائى فى الذاكرة .

LOAD < finary filename >

86 - LOCATE

يستخدم للبحث عن أول سجل يقابل شرط محدد فى ملف قاعدة بيانات مفتوح

LOCATE < cond > [< scope >] [WHILE < cond >]

87 - MODIFY COMMAND

يستخدم لعرض شاشة تنقيح dBase IV بغرض إنشاء أو تعديل ملف برنامج أو ملف تشكيل أو حقول ذاكرة أو ملف نصى (text)

MODIFY COMMAND < filename > [WINDOW
< window name >]

88 - MOVE WINDOW

يستخدم لنقل نافذة الى موقع آخر على الشاشة .

MOVE WINDOW < window name > TO < r > , < c > /
BY (row offset > , < column offset >

89 - NOTE / * / &&

يستخدموا لكتابة بعض التعليقات فى البرنامج .

NOTE / * / && < text >

90 - ON ERROR / ESCAPE / KEY

يستخدم لترقب حدوث أحد ثلاث شروط إما حدوث خطأ ERROR أو الضغط على مفتاح ESCAPE أو الضغط على مفتاح محدد KEY ثم يقوم بتنفيذ برنامج أو إجراء فرعى .

ON ERROR / ESSCAPE [< commands >]
ON KEY [< key label name >] [< commands >]

91 - ON PAD

يستخدم لتنفيذ قائمة رأسية عند وجود المؤشر على أحد اختيارات القائمة الأفقية .

ON PAD < pad name > OF < menu name >
ACTIVATE POPUP < popup name >

92 - ON PAGE

يستخدم فى التحكم فى طباعة رأس وتذييل صفحات التقرير .

ON PAGE [AT line < exp N > < command >

93 - ON READ ERROR

يستخدم لترقب حدوث أى أخطاء أثناء استخدام شاشة كاملة [أى الشاشة التى تستخدم الأوامر :

@ ..... say ..... get
read

ويقوم بتنفيذ أحد الأوامر أو أحد الاجراءات أو أحد البرامج الفرعية .

ON READ ERROR [< command >]

94 - ON SELECTION PAD

يستخدم لتنفيذ أمر محدد أو إجراء أو برنامج عند اختيار أحد اختيارات القائمة الأفقية .

ON selection PAD < pad name > OF < menu name >
[< command >]

95 - ON SELECTION POPUP

يستخدم لتحديد أحد الأوامر ليتم تنفيذه عند الاختيار من قائمة رأسية .

ON SELECTION POPUP < popup name > / ALL [BLANK]
[< command >]

96 - PACK

يستخدم لحذف كل السجلات السابق تمييزها باستخدام الأمر delete من ملف قاعدة البيانات المفتوح .

PACK

97 - PARAMETERS

يستخدم لتحديد قيم لحقول ذاكرة متغيرة تسمى كشف المعطيات (parameter list) ويتم تمرير هذه القيم الى برنامج باستخدام الأمر :

DO .... WITH < parameter list >

98 - PLAY MACRO

يستخدم لتنفيذ عملية مجمعة (MACRO) .

PLAY MACRO < macro name >

99 - PRINT JOB / END PRINT JOB

يستخدم للتحكم فى تنفيذ الأوضاع المرتبطة بعملية الطباعة .

PRINT JOB
< commands >
ENDPRINT JOB

100 - PRIVATE

يستخدم لانشاء حقول ذاكرة متغيرة أو عناصر مصفوفة لها نفس أسماء حقول ذاكرة عناصر مصفوفة سبق إعلانها (PUBLIC) دون أن تتأثر محتوياتها بها .

PRIVATE [< memvar list > / ARRAY < array definition list >] [ALL [like / EXCEPT < skelton >]]

101 - PROCEDURE

يستخدم للتعريف ببداية إجراء .

PROCEDURE < procedure name >

102 - PROTECT

يستخدم للحفاظ على سرية البيانات ولإنشاء نظام لتأمين البيانات .

103 - PUBLIC

يستخدم لإنشاء حقول ذاكرة متغيرة أو عناصر مصفوفة يمكن استخدامها أو تعديلها فى الاجراءات أو البرامج المتفرعة .

PUBLIC < memory variable list > / [ARRAY < array definition list >]

104 - QUIT

يغلق جميع الملفات ويتم الانتقال الى نظام التشغيل DOS .

105 - READ

يستخدم لتنفيذ كل أوامر GET ..... @ قبله .

106 - RECALL

يستخدم لإزالة العلامات المميزة كنتيجة لإستخدام الأمر delete فى ملف قاعدة البيانات المفتوح .

RECALL [< scop >] [FOR < condition >] [WHILE < condition >]

107 - REINDEX

يستخدم لإعادة بناء كل الفهارس فى ملف قاعدة البيانات المفتوح .

108 - RELEASE

يستخدم لحذف حقول الذاكرة المتغيرة وبرامج لغة التجميع (assemply) والقوائم الأفقية والرأسية والنوافذ من الذاكرة .

RELASE < memvar list > / ALL [like / except / < sbeleton >]
RELASE MODULE < module name list > / Menus < menu
< name list > / POPUPS < poup name list > / WINDOW
< window name list >

109 - RENAME

يستخدم لتغيير أسم ملف على القرص الممغنط .

RENAME < old filename > TO < new filename >

110 - REPLACE

يستخدم لتغيير محتويات حقل فى ملف قاعدة بيانات مفتوح .

REPLACE < filed name 1 > WITH < exp 1 > [, < field name 2 >
WITH < exp 2 ..] [< scop >] [FOR < cond >] [WHILE < cond >]
[ADDITIVE]

111 - REPORT FORM

يستخدم لتوجيه خرج ملف التقرير للشاشة أو الطابعة أو ملف نصى (text)

REPORT FORM < filename > / [PLAIN] [HEADING < exp c >]
[NOEJECT] [SUMMARY] [< scop >] [FOR < cond >]
[WHILE < cond >] [TO PRINTER / TO FILE < filename >]

112 - RESTORE

يستخدم لتحميل حقول الذاكرة المتغيرة والمصفوفات من ملف ذاكرة (.MEM) فى الذاكرة .

RESTORE FROM < filename > [ADDITIVE]

أ - RESTORE FROM < filename >

أولاً يحذف كل حقول الذاكرة والمصفوفات من الذاكرة ثم يحمل حقول الذاكرة والمصفوفات من الملف الى الذاكرة .

ب - RESTORE FROM < filename > ADDITIVE

يضيف حقول الذاكرة والمصفوفات من الملف الى الذاكرة .

113 - RESTORE MACROS

يستخدم لتحميل عمليات مجمعة (سبق حفظها) فى الذاكرة .

RESTORE MACROS FROM < macro file >

114 - RESTORE WINDOW

يستخدم لتحميل ذاكرة أو أكثر من القرص الممغنط الى الذاكرة .

RESTORE WINDOW < window name list > / ALL FROM < filename >

115 - RESUM

يستخدم لمواصلة تنفيذ برنامج سبق وقفه باستخدام الأمر SUSPEND .

116 - RETRY

يستخدم لإعادة تنفيذ أمر سبق حدوث خطأ أثناء عملية تنفيذه .

117 - RETURN

يستخدم للإنتقال إما للبرنامج المستدعى أو نقطة الأوامر .

RETURN [< exp >] [TO MASTER / TO < procedure name >]

118 - ROLLBACK

يستخدم لإلغاء أى تغيرات أحدثت فى ملفات قاعدة البيانات والفهارس أثناء عملية التعديل .

ROLLBACK [< data base filename >]

119 - RUN / !

يستخدم لتنفيذ أحد أوامر نظام التشغيل

RUN / ! < DOS command >

120 - SAVE

يستخدم لحفظ حقول الذاكرة المتغيرة وعناصر الذاكرة في ملف .

SAVE TO < filename > [ALL like / EXCEPT < skelton >]

121 - SAVE MACROS

يستخدم لحفظ العمليات المجمعة الحالية على القرص الممغنط .

SAVE MACROS TO < macro file >

122 - SAVE WINDOW

يستخدم لحفظ تعريف نافذة أو أكثر على القرص الممغنط .

SAVE WINDOW < window name list > / ALL TO < filename >

123 - SCAN

يقوم بمسح ملف قاعدة البيانات ويختار سجلات محددة لإجراء بعض العمليات عليها .

```
SCAN [ < scope > ] [ FOR < condition > ] [ WHILE < condition > }
      [ < commands > ]
ENDSCAN
```

124 - SEEK

يستخدم للبحث عن البيانات فى الملف المفهرس .

SEEK < exp >

125 - SELECT

يستخدم لاختيار احدى مناطق الذاكرة من (١) الى (١٠) أو من (A) الى (J)

126 - SHOW MENU

يستخدم لإظهار قائمة أفقية على الشاشة دون أن تكون هذه القائمة فعالة .

```
SHOW MENU < menu name > [ PAD < pad name > ]
```

127 - SKIP

تستخدم لتغيير موقع مؤشر السجلات فى الملف .

```
SKIP [ < exp > ] [ IN < alias name > ]
```

128 - SORT

يستخدم لإنشاء ملف جديد من كل سجلات ملف قاعدة بيانات مفتوح (أو بعضاً منه) مرتباً ترتيباً معيناً .

```
SORT  TO < filename > ON < field > [ / A ] [ / C ] [ / D ]
[ , < field 2 > [ / A ] [ / C ] [ / D ] ... ]
[ ASCENDING / DESCENDING ] [ scope > ]
[ ROR < cond > ] [ WHILE < cond > ]
```

129 - STORE

يستخدم لإنشاء حقول ذاكرة متغيرة أو ملئ عناصر مصفوفة .

```
STOR [ < exp > TO < memvar list > / < array element list >
```

130 - SUM

يستخدم لحساب حاصل جمع تعبيرات رقمية ويقوم بتخزين الناتج فى حقل ذاكرة متغير أو عنصر مصفوفة .

```
SUM < exp N list > ] [ TO < memvar list > / TO ARRAY
< array name > ] [ < scope > ] [ FOR < cond > ]  [ WHILE
< cond > ]
```

131 - SUSPEND

يستخدم لتعطيل تنفيذ البرنامج بصفة مؤقتة لإتاحة فرصة لإدخال أوامر باستخدام لوحة المفاتيح .

132 - TEXT

يستخدم لإظهار نص على الشاشة أو طباعته على الطابعة .

TEXT < text character > ENDTEXT

133 - TOTAL

يستخدم لجمع حقول رقمية فى ملف قاعدة بيانات مفتوح ويقوم بإنشاء ملف قاعدة بيانات للاحتفاظ بنتيجة علمية الجمع .

TOTAT ON < key field > TO < filename > [FIELDS < field list >] [< scope >] [FOR < cond >] [WHILE < cond >]

134 - TYPE

يستخدم لإظهار محتويات ملف على الشاشة أو طباعته على الطابعة .

TYPE < filename > [NUMBER] [TO PRINTER / TO FILE < filename >]

أ - NUMBER لإظهار رقم السطر مع كل سطر فى الملف .
ب - TO PRINTER لإرسال الخرج الى الطابعة .
ج - TO FILE لإرسال الخرج الى ملف .

135 - UPDATE

يستخدم لإستبدال محتويات حقول فى ملف قاعدة بيانات مفتوح ببيانات من ملف قاعدة بيانات آخر .

UPDATE ON < key field > FROM < alias name > REPLACE < field name > WITH <exp 1> [,< field name 2 > WITH < exp 2 > ..] RANDOM

136 - USE

يستخدم لفتح ملف قاعدة بيانات والفهرس وملف الذاكرة المرتبطة به .

```
USE < database filename > / ? ] [ IN < worke area  number >  ]
[ [ INDEX < . ndx or . mdx file list > ]
[ ORDER [ TAG ] < . ndx filename > / < . mdx tag >
[ OF < . mdx filename > ] ] [ ALIAS < alias > ]
[ EXCLUSIVE ] [ NOUPDATE ] ]
```

137 - WAIT

يستخدم لإيقاف تنفيذ البرنامج بصفة مؤقتة حتى يتم ضغط أى مفتاح .

```
WAIT [ < prompt > ] [ TO < memvar > ]
```

138 - ZAP

يستخدم لحذف كل سجلات ملف قاعدة بيانات مفتوح .

ثانياً : SET COMMANDS

1 - SET ALTERNATE

الشكل العام :

SET ALTERNATE TO [< filename] [ADDITIVE]
SET ALTERNATE ON / OFF

يستخدم لإرسال نتيجة الأوامر التالية له لملف نصى .

2 - SET AUTOSAVE

الشكل العام :

SET AUTOSAVE ON / OFF

الوضع المسبق SET AUTOSAVE OFF

يستخدم لحفظ كل السجلات بداخل الملف بعد كل اضافة أو تعديل للسجل .

3 - SET BELL

الشكل العام :

SET BELL ON / OFF
SET BELL TO [< frequency > , < duration >]

الوضع المسبق SET BELL TO 512,

يستخدم لتشغيل أو عدم تشغيل سماعة (speaker) الحاسب ، وكذلك للتحكم فى تنغيم الصوت .

4 - SET BLOCKSIZE

الشكل العام :

SET BLOCKSIZE TO < exp N >

الوضع المسبق SET BLOCKSIZE TO 1
N = 1 TO 32

يستخدم لضبط عدد مواقع الذاكرة المخصصة لحقول الذاكرة memo fields) وملفات الفهرسة المتعددة (. mdx) .

5 - SET BORDER

الشكل العام :

SET BORDER TO [single / double / panel / none / border definition string]

يستخدم لتغيير طرز أحرف الإطارات (Boxes) .

6 - SET CARRY

الشكل العام :

SET CARRY ON / OFF
SET CARRY TO [< field list >] [ADDITIVE]

الوضع المسبق : SET CARRY OFF

ينسخ كل حقول (أو حقول مختارة) سجل لسجل تالى فى حالة استخدام أى من الأوامر التالية INSERT , APPEND

7 - SET CATALOGE

الشكل العام :

SET CATALOG ON / OFF
SET CATALOG TO [< filename >] [?]

الوضع المسبق : SET CATALOG TO OFF

يستخدم للتحديث التلقائى فى ملف كتالوج عند استخدام أى من الأوامر التالية
CREATE , INDEX , CREATE / MODIFY REPORT

8 - SET CENTURY

الشكل العام : SET CENTURY ON / OFF

الوضع المسبق : SET CENTURY OFF

يستخدم لعرض القرن ضمن تعبير تاريخى (أى يكون الرقم المعبر عن السنة أربعة أرقام) .

9 - SET CLOCK

الشكل العام :

SET CLOCK ON / OFF
SET CLOCK TO [< row , column >]

الوضع المسبق : SET CLOCK OFF

يستخدم لعرض الساعة (حسب ضبطها فى نظام التشغيل) فى موقع محدد على الشاشة .

10 - SET COLOR

الشكل العام :

SET COLOR ON / OFF
SET COLOR TO [[< standard >] [, < enhanced >]
[, < perimeter >] [, < background >]]

يستخدم لضبط الألوان .

11 - SET CONFIRM

الشكل العام : SET CONFIRM ON / OFF

الوضع المسبق : SET CONFIRM OFF

يستخدم لتحديد إذا كان المؤشر سوف يتحرك تلقائياً للحقل التالى فى حالة إمتلاء الحقل الحالى أم لا .

12 - SET CONSOLE

الشكل العام : SET CONSOLE ON / OFF

الوضع المسبق : SET CONSOLE ON

يستخدم للتحكم فى عرض أو عدم عرض التقارير على الشاشة فى نفس وقت طبعها على الطابعة .

13 - SET DATE

الشكل العام : SET DATE < format >

الوضع المسبق : SET DATE AMERICAN

يستخدم لضبط شكل إظهار التاريخ .

14 - SET DEBUG

الشكل العام : SET DEBUG ON / OFF

الوضع المسبق : SET DEBUG OFF

يستخدم للوصول الى الأخطاء الموجودة بالبرنامج مع عرضها أو طبعها على الطابعة .

15 - SET DECIMALS

الشكل العام : SET DECIMALS TO < exp N >

الوضع المسبق : SET DECIMALS TO 2

يستخدم لتحديد عدد مواقع الكسر العشرى .

16 - SET DEFAULT

الشكل العام : SET DEFAULT TO < drive >

الوضع المسبق : SET DEFAULT TO C

يستخدم لاختيار مشغل وحدة آخر .

17 - SET DELETED

الشكل العام : SET DELETED ON / OFF

الوضع المسبق : SET DELETED OFF

يستخدم لتحديد أسلوب التعامل مع السجلات المختارة للحذف .

SET DELETED ON تعنى إهمال السجلات المختارة للحذف عند تنفيد أوامر فيما عدا أمرى RECALL ALL , GOTO .

18 - SET DELIMITERS

الشكل العام :

SET DELIMITERS ON / OFF
SET DELIMITERS TO < exp c > / DEFAULT

الوضع المسبق : SET DELIMITERS OFF

يستخدم لتحديد شكل بداية ونهاية الحقول .

19 - SET DEVELOPMENT

الشكل العام : SET DEVELOPMENT ON / OFF

الوضع المسبق : SET DEVELOPMENT OFF

يستخدم لتحويل البرنامج المصدر (prg .) (source program) تلقائياً الى (object program) أو (dbo .) بعد كل تعديل قبل استخراج البرنامج .

20 - SET DEVICE

الشكل العام : SET DEVICE TO SCREEN / PRINTER / FILE < filename >

الوضع المسبق : SET DEVICE TO SCREEN

يستخدم لتحديد الجهة التى سيرسل اليها نتيجة الأمر say ..... @ وهى إما الطابعة أو الشاشة أو ملف .

21 - SET DISPLAY TO

الشكل العام : SET DISPLAY TO MONO / COLOR / EGA 25 / EGA 43 / MONO 43 / VGA 25

يستخدم لاختيار نوع الشاشة وتحديد عدد الأسطر الأفقية للشاشة .

22 - SET ECHO

الشكل العام : SET ECHO ON / OFF

الوضع المسبق : SET ECHO OFF

يستخدم لعرض أسطر الأوامر أثناء تنفيذها .

23 - SET ESCAPE

الشكل العام : SET ESCAPE ON / OFF

الوضع المسبق : SET ESCAPE ON

يستخدم لتحديد إذا ما كان سيتم إهمال ضغط مفتاح (ESC) أم لا .

24 - SET EXACT

الشكل العام : SET EXACT ON / OFF

الوضع المسبق : SET EXACT OFF

يستخـــدم لتحديد إن كان سيتم تطبيق شروط التطابق التام عند مقارنة تعبيرين أم لا .

25 - SET FIELDS

الشكل العام : 1 - SET FIELDS ON / OFF

الوضع المسبق : SET FIELDS OFF

عندما تكون فى الوضع (ON) الحقول المختارة فقط هى التى يمكن الوصول اليها أو عرضها .

1 - SET FIELDS TO < field > [/R] / < calculated field id > .....] [, < field > [/ R] / < calculated field id > ....] [/ ALL [like / except < skeleton >]

تستخدم لتحديد كشف بالحقول التى يمكن الوصول اليها أو عرضها فى ملف واحد أو أكثر من ملف .

/ R تعنى أن هذه الحقول للقراءة فقط (Read only) .

26 - SET FILTER

الشكل العام : SET FILTER TO [file < filename > / ?] [< condition >]

تستخدم لقصر الوصول الى سجلات محددة من ملف قاعدة البيانات والتى تقابل شرط معين . وذلك لغرض إجراء عمليات معينة على هذه السجلات فقط دون باقى سجلات الملف .

27 - SET FORMAT

الشكل العام : SET FORMAT TO [< format filename > / ?]

تستخدم لفتح ملف تشكيل (FORMAT FILE) .

28 - SET FUNCTION

الشكل العام : SET FUNCTION < exp N > TO < exp C >

تستخدم لتغيير وظيفة أحد مفاتيح الوظائف (من F1 الى F10) .

29 - SET INDEX

الشكل العام :

SET INDEX TO [? / [< filename list > [ORDER [TAG] < .ndx filename > / < TAGNAME > [OF < .mdx filename >]]]]

تفتح كشف محدد من ملفات الفهرسة أو ملفات الفهرسة المتعددة وتحدد الفهرس المتحكم .

30 - SET HEADING

الشكل العام : SET HEADING ON / OFF

الوضع المسبق : SET HEADING ON

تستخدم للتحكم فى إظهار أو عدم إظهار عناوين الأعمدة عند استخدام الأوامر :

SUM , LIST , DISPLAY , AVERAGE

31 - SET HELP

الشكل العام : SET HELP ON / OFF

الوضع المسبق : SET HELP ON

تستخدم للتحكم فى إظهار أو عدم إظهار بيانات المساعدة (HELP) عند حدوث خطأ فى كتابة أحد الأوامر .

32 - SET HISTORY

الشكل العام : SET HISTORY ON / OFF

الوضع المسبق : SET HISTORY ON

يوجد فى dBase IV وسيط خازن يسمى (history buffer) يخزن حتى ٢٠ أمر سبق تنفيذهم من خلال نقطة الأوامر ، وهذا الأمر يتحكم فى إمكانية تخزين الأوامر السابق تنفيذها فى هذا الوسيط الخازن أم لا .

SET HISTORY TO < exp N >

تتحكم فى تحديد عدد الأوامر السابق تنفيذها والممكن تخزينها فى الوسيط الخازن .

33 - SET INTENSITY

الشكل العام : SET INTENSITY ON / OFF

الوضع المسبق : SET INTENSITY ON

تستخدم لعكس الألوان عند إستخدام الأمر (أو لعكس الإضاءة فى الشاشة العادية)

@ .... GET

لتصبح تماماً كما هو الحال عند تنفيذ الأمر :

@ .... SAY

34 - SET MARK

الشكل العام : SET MARK TO [< exp C >]

تستخدم لتنفيذ الفواصل بين اليوم والشهر والسنة عند إظهار التاريخ .

35 - SET MARGIN

الشكل العام : SET MARGIN TO [< exp N >]

الوضع المسبق : SET MARGIN TO 0

تستخدم لضبط الهامش الأيسر فى الطباعة .

36 - SET MEMOWIDTH

الشكل العام : SET MEMOWIDTH TO [< exp N >]

الوضع المسبق : SET MEMOWIDTH TO 50

يستخدم لضبط العرض (WIDTH) لحقل الذاكرة (MEMO FIELD) أقل عرض هو ٨ وأكبر عرض هو ٢٥٥ .

37 - SET MESSAGE

الشكل العام : SET MESSAGE TO [< exp N >]

يستخدم لعرض جملة فى السطر الأخير للشاشة فى حالة أن يكون :
SET STATUS فى الوضع ON

38 - SET TITLE ON / OFF

يستخدم لإظهار أو عدم إظهار السؤال عن أسم ملف الكتالوج عند إضافة ملف جديد لكتالوج .

39 - SET NEAR ON / OFF

الشكل العام : SET NEAR ON / OFF

يستخدم لوضع مؤشر السجلات لأقرب سجل يقارب شرط البحث عند عدم تحقق شرط البحث بإستخدام أمرى : SEEK , FIND

40 - SET ORDER

الشكل العام : SET ORDER TO [< exp N >] / [TAG < mdx tag > [OF < mdx name >]]

يستخدم لفتح فهرس معين من النوع (.mdx).

SET ORDER TO

يستخدم لغلق كل الفهارس وإعادة الملف لترتيب السجلات الأصلى .

41 - SET PATH

الشكل العام : SET PATH TO [< path list >]

يستخدم لتحديد دليل آخر غير الدليل الحالى .

42 - SET POINT

الشكل العام : SET POINT TO [< path list >]
الوضع المسبق : SET POINT TO " , "

يستخدم لتغيير شكل العلامة المستخدمة فى الأرقام العشرية .

43 - SET PROCEDURE

الشكل العام : SET PROCEDURE TO [< procedure filename>]

يستخدم لفتح ملف إجراءات .

44 - SET PRINTER

الشكل العام : SET PRINTER ON / OFF
الوضع المسبق : SET PRINTER OFF

يستخدم لتوجيه المخرجات الى الطابعة أو الى ملف .

ملحوظة : المخرجات التى لا يستخدم فيها أمر : @ ..... say
على عكس إستخدام أمر : SET DEVICE TO PRINTER

45 - SET RELATION

الشكل العام : SET RELATION TO < exp > / < exp N > INTO < alias >
[, < exp N > INTO < alias >] ...
يستخدم لربط ملفين أو أكثر تبعاً لمفتاح مشترك لجميع الملفات .

46 - SET STATUS

الشكل العام : SET STATUS ON / OFF

الوضع المسبق : SET STATUS ON

يستخدم لإظهار أو عدم إظهار سطر الحالة أسفل الشاشة .

47 - SET SAFETY

الشكل العام : SET SAFETY ON / OFF

الوضع المسبق : SET SAFETY ON

يستخدم لإظهار أو عدم إظهار رسالة التحذير التالية :

FILE ALREADY EXISTS .. OVERWRITE ? (Y / N)

عند إنشاء ملف جديد له نفس أسم ملف موجود فعلاً .

48 - SET SCOREBOARD

الشكل العام : SET SCOREBOARD ON / OFF

الوضع المسبق : SET SCOREBOARD ON

يستخدم لإظهار أو عدم إظهار رسالة فى السطر الأول للشاشة فى حالة إستخدام أمر : DELETE عندما تكون : SET STATUS OFF يتم إظهار الرسالة . وعندما تكون : SET STATUS ON لا يتم إظهار الرسالة فى السطر الأول من الشاشة .

49 - SET SEPERATOR

الشكل العام : SET SEPERATOR TO [< exp C >]

يستخدم لتغيير الفاصلة المستخدمة كعلامة فاصلة فى الأرقام لأى علامة أخرى .

50 - SET SKIP

الشكل العام : SET SKIP TO [< alias > [, < alias >] ...]

عند ربط ملفين أو أكثر بإستخدام الأمر : SET RELATION فإن أمر : SET SKIP يسمح بالوصول الى السجلات التالية والتى لها نفس المفتاح فى الملفات المترابطة .

51 - SET STEP

الشكل العام : SET STEP ON / OFF
الوضع المسبق : SET STEP OFF

يستخدم لإيقاف تنفيذ البرنامج بعد تنفيذ كل سطر فى البرنامج ، ويستخدم للتأكد من خلو البرنامج من الأخطاء .

52 - SET TALK

الشكل العام : SET TALK ON / OFF
الوضع المسبق : SET TALK ON

يستخدم لإظهار أو عدم إظهار نتيجة تنفيذ بعض الأوامر مثل :
SUM , STORE , PACK , COPY .

53 - SET TYPEAHEAD

الشكل العام : SET TYPEAHEAD TO < exp N >
الوضع المسبق : SET TYPEAHEAD TO 20

يستخدم لتحديد وسيط الكتابة الخازن (TYPEAHEAD BUFFER) أقصى عدد مسموح به هو ٣٢٠٠٠ حرف .

54 - SET UNIQUE

الشكل العام : SET UNIQUE ON / OFF

الوضع المسبق : SET UNIQUE OFF

يستخدم لإضافة السجلات المكررة (أى التى لها نفس المفتاح) إلى ملف الفهرسة أو عدم إضافتها .

55 - SET VIEW

الشكل العام : SET VIEW TO < query filename > / ?

يستخدم لفتح ملف إستفسار .

56 - SET WINDOW

الشكل العام : SET WINDOW OF MEMO TO < window name >

يستخدم لفتح نافذة سبق تعريفها بغرض تنقيح أو تعديل حقل ذاكرة (memo field)

*****

فهرس الجزء الثالث

فهرس الجزء الثالث

رقم الصفحة

● الفصل التاسع

● الفصل العاشر

● الفصل الحادى عشر

REFERENCES

* dBASE IV Advanced Topics, ASHTON TATE, 1988.
* dBASE IV LANGUAGE REFERENCE, ASHTON TATE, 1988.
* dBASE IV GETTING STARTED with dBASE IV, ASHTON TATE, 1988.
* MASTERING dBASE IV, CARL TAVENSEND. BPB BOOK CENTRE, INDIA, 1989.
* AUTOCAD REFERENCE MANUAL. Autodesk Comp.
* dBASE IV Version 1.1 ASHTON TATE, 1991.

* * * * *

رقم الايداع بدار الكتب
٢٣٨٠ / ١٩٩١
الترقيم الدولى
977 - ٥٠٣٥-٠٤-X

مزايا . الاصدار الجديد (١-١)

١ - يشغل حيزا اقل فى القرص الصلب من الاصدار السابق .

٢ - احتياج اقل لمقدار الذاكره حيث تم تقليل مقدار الذاكره المطلوبه من ٥١٦ كيلو بايت الى ٤٥٠ كيلو بايت .

٣ - امكن تطبيق نظام جديد لادارة الذاكره الديناميكيه (dmms)الذى تسبب فى تقليل الحاجة للوصول الى القرص الصلب اثناء العمل مع البرنامج مما تسبب عنه زياده سرعه الاداء .

٤ - اعطى تسهيلات اكثر فى امكانية استخدام مقدار الذاكره الزائد عن ٦٤٠ ك بايت .

٥- أكثر سهولة فى مواءمته مع الحاسبات الشخصية المتوافقة مع IBM .

٦ - تم اختبار الاصدار الجديد بواسطة اكثر من ٢٧٠٠ خبير .

5035 - 04 - x

دار الكتب العلمية
للنشر والتوزيع
١٨ شارع السبع - ترعة السواحل إمبابه